UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

20|21

Trabajo fin de grado

Diseño e implementación de un sistema de análisis de colas

Sergio García Bustos

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en ingeniería informática

TRABAJO DE FIN DE GRADO

Diseño e implementación de un sistema de análisis de colas

Sergio García Bustos

Tutor: Álvaro Ortigosa Juárez

JUNIO 2021

Sergio García Bustos
Diseño e implementación de un sistema de análisis de colas

Sergio García Bustos
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

AGRADECIMIENTOS

A mi familia por todo el apoyo que me han dado en todo momento para permitirme seguir estudiando.

A mis amigos por ayudarme tanto en los estudios como fuera de ellos en los momentos mas complicados.

A los profesores del Grado de Ingeniería Informática de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid que me han enseñado todo lo necesario para llegar a dónde estoy ahora.

A mi tutor Álvaro por proponer este TFG y ayudarme a terminarlo.

Resumen

Este Trabajo Fin de Grado tenía como objetivo principal el desarrollar una herramienta para la estimación del rendimiento de sistemas informáticos distribuidos basada en la simulación de eventos discretos. A la hora de estimar el rendimiento existen varias opciones entre las que elegir, cada una con un coste y una precisión asociada, y en esta carrera hemos visto un método de coste bajo para estimar el rendimiento mediante modelos matemáticos simples. Sin embargo, para que se puedan aplicar estos modelos hay que hacer primero una serie de simplificaciones, lo cual baja la precisión de la estimación. Otros métodos más precisos consisten en construir sistemas con características similares a modo de prototipos, pero estos métodos, además del coste adicional que requiere montar un sistema prototipo, carecen de flexibilidad, ya que si se quisiera modificar el sistema para probar cómo afectaría un cambio al rendimiento se tendría que modificar el prototipo aumentando el coste de la estimación.

Aquí entra la simulación, como un método de estimación de rendimiento intermedio que ofrece la flexibilidad en las modificaciones de los modelos matemáticos simples sin tener tantas simplificaciones, aumentando la precisión de la estimación sin aumentar el coste tanto como lo haría el construir un prototipo.

Para este proyecto se ha usado primero lenguajes conocidos por el programador, como Python, para construir el simulador y luego se ha rescrito el programa en Javascript para poder crear una aplicación Web que permita descargar en el navegador la aplicación, simplificando la instalación, ya que solo debería instalarse una vez en el servidor para que cualquiera pueda usarla, y para añadir una interfaz gráfica basada en grafos para construir y modificar el sistema que se va a simular.

Como resultado se ha desarrollado una aplicación accesible, que permite construir un modelo del sistema informático a simular y modificarlo sin dificultad y que no requiere de amplios conocimientos sobre los modelos matemáticos de los sistemas informáticos distribuidos, aunque si se requieren conocimientos sobre sus parámetros y sobre los propios sistemas informáticos distribuidos para analizar el resultado de la simulación.

Palabras clave

Sistemas informáticos distribuidos, teoría de colas, React, Javascript, aplicación Web Simulación

Abstract

The main objective of this Final Degree Project was to develop a tool for performance estimation of distributed computing systems based on discrete event simulation. When estimating performance there are several options to choose from, each with an associated cost and accuracy, and in this course we have seen a low-cost method for estimating performance using simple mathematical models. However, in order for these models to be applied, a number of simplifications must first be made, which lowers the accuracy of the estimate. Other more accurate methods consist of building systems with similar characteristics as prototypes, but these methods, in addition to the additional cost required to assemble a prototype system, lack flexibility, since if the system were to be modified to test how a change would affect performance, the prototype would have to be modified, increasing the cost of the estimate.

This is where simulation comes in, as an intermediate performance estimation method which offers flexibility in the modifications of the simple mathematical models without having so many simplifications, increasing the accuracy of the estimation without increasing the cost as much as building a prototype would.

For this project we first used languages known to the programmer, such as Python, to build the simulator and then rewrote the program in Javascript in order to create a Web application that allows the application to be downloaded to the browser, simplifying installation, since it should only be installed once on the server so anyone can use it, and to add a graph-based graphical interface to build and modify the system to be simulated.

As a result, an accessible application has been developed, which allows building a model of the computer system to be simulated and modifying it without difficulty and which does not require extensive knowledge of the mathematical models of distributed computer systems, although knowledge of their parameters and of the distributed computer systems themselves is required to analyze the result of the simulation.

Keywords

Distributed computing systems, queueing theory, React, Javascript, web application Simulation

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1

INTRODUCCIÓN

1.1. Motivaciones

La motivación principal de este proyecto consiste en crear una herramienta para la estimación del rendimiento de los Sistemas Informáticos Distribuidos basada en la simulación de eventos discretos [1] con la que se puedan evaluar tanto los sistemas analizables por modelos matemáticos simples [2] como algunos de los sistemas que no lo son. Por tanto, la herramienta tendrá un doble uso como apoyo para el aprendizaje de los modelos matemáticos simples y como método para estimar el rendimiento de los sistemas que no son analizables por dichos modelos.

Cuando analizamos un sistema informático, ya sea como aprendizaje, para evaluar costes o su viabilidad, es fundamental la evaluación de sus requisitos.

Dichos requisitos se pueden dividir en dos categorías:

- Funcionales: aquellos que definen el comportamiento lógico del sistema y se centran en el problema a resolver por el sistema [3].

- No funcionales: aquellos que definen el comportamiento físico del sistema y se centran en el modo en el que el sistema resuelve el problema, así como las limitaciones y factores a tener en cuenta a la hora de resolverlo [4].

Los requisitos funcionales afectan al diseño de un Sistema Informático Distribuido de manera parecida a como afectarían a un sistema de un único ordenador, restringiendo la elección del modelo a utilizar y favoreciendo a unos paradigmas frente a otros. Mientras que los requisitos no funcionales afectarían en mayor escala al diseño global del sistema dado que limitan el comportamiento del sistema entero, por ejemplo, los requisitos no funcionales de rendimiento.

Entre los requisitos no funcionales estarían los requisitos de rendimiento, que son aquellos que definen los requisitos de disponibilidad temporal del sistema, no solo en la duración máxima aceptable del procesamiento de una petición el sistema, sino que también se incluyen otros parámetros como [5]:

- Latencia: Tiempo de respuesta del sistema frente a un evento.

- Productividad: Número de respuestas a eventos que el sistema es capaz de realizar por unidad de tiempo en un intervalo determinado.

- Capacidad: Cantidad máxima de trabajo que el sistema puede realizar, puede estar limitado por una latencia máxima.

Cada uno de los parámetros arriba mencionados requiere un análisis específico como parte del diseño del sistema, pero, además, se debe tener en cuenta como cada uno influye en los demás. Por ejemplo, la latencia en las llegadas a un componente puede limitar la capacidad usada en el mismo, pudiendo potencialmente desaprovechar capacidad.

En conclusión, la estimación del rendimiento de un Sistema Informático Distribuido no sería algo intuitivo o fácilmente medible con una precisión aceptable y/o un coste asequible, de hecho, se puede dividir los métodos en grupos según estas dos escalas: precisión y coste [6].

Es concebible que, si estamos llevando a cabo una evaluación de rendimiento de un sistema informático distribuido, cuanta mayor precisión requiramos (efectividad/eficiencia conseguida), mayor sea el coste asociado (volumen de recursos requeridos). Por este motivo, la elaboración de una escala donde podamos ubicar los diferentes métodos de estimación utilizando los parámetros mencionados, nos ayudará a observar más claramente la situación.

Cada método se asociará a un rango estimado de precisión y de coste, que, irá aumentando progresivamente, obteniendo diferentes grupos localizados en diferentes tramos de la escala.

Dichos grupos son, en orden ascendente de precisión y coste:

- Estimación “a ojo” (valoración por un experto): Este método requiere que la evaluación la lleve a cabo una persona con sobrada experiencia y conocimientos profundos del tipo de sistema a analizar. La precisión de los resultados dependerá de la complejidad del sistema analizado. Este método es normalmente utilizado para descartar, de una manera rápida y utilizando pocos recursos, requerimientos que puedan exceder el rendimiento realmente requerido o no alcancen las expectativas impuestas.

- Estimación por modelos matemáticos simples: Este método requiere que la evaluación la lleve a cabo una persona con conocimientos necesarios sobre los modelos matemáticos simples para poder aplicarlos. Este método debería de mejorar la precisión de la estimación con respecto al método anterior, y conlleva la realización de una serie de simplificaciones al sistema a analizar que limitarán su precisión. Este método no sería aplicable a todos los sistemas, solo a aquellos que admitan el desarrollo de modelos matemáticos simples.

- Estimación por simulación: Este método reduce los requerimientos de simplificación del anterior, aumentando, por tanto, su precisión. También se aumenta el coste dado que requerirá más recursos como la creación o uso de herramientas para llevar a cabo las simulaciones.

- Estimación por prototipos: Este método alcanza un nivel de precisión bastante alto, ya que incluye diseño de sistemas físicos cercanos a diseño requerido. El coste se incrementa al aumentar los recursos necesarios. La factibilidad de aplicar este método depende directamente de la limitación de nuestros recursos. Cada nuevo sistema para analizar requerirá el diseño y la construcción de un prototipo del mismo.

- Mediciones en el sistema real: Este método tiene la precisión más alta posible dado que las medidas son recogidas del propio sistema real. Este método no se anticipa a los posibles fallos, solo nos muestra los aciertos o fallos de nuestra decisión.

De todos los métodos mencionados, los modelos matemáticos simples fueron comentados en la asignatura Sistemas Informáticos II de Ing. Informática, EPS-UAM [7], donde se mostró el modo de utilización de estos modelos para llevar a cabo las estimaciones de rendimiento, permitiendo que el estudiante comprenda los detalles subyacentes en los análisis y entienda cómo se llega a las ecuaciones que se utilizan en los distintos escenarios, así como explicar el alcance y limitaciones de este método. En esta línea, resultaría de suma utilidad la existencia de una herramienta que permita contrastar los análisis teóricos y, por otra parte, extender el análisis más allá de los métodos matemáticos, relativamente simples, permitiendo simulaciones de diferentes escenarios.

Este proyecto pretende crear una herramienta de estimación del rendimiento de sistemas informáticos distribuidos basada en la simulación de eventos discretos. Mediante este método se mantendrá un coste bajo (pocos recursos), se aumentará la flexibilidad para analizar distintos sistemas/escenarios y se aumentará la precisión eliminando o reduciendo las simplificaciones asociadas a la aplicación de modelos matemáticos simples. A diferencia de los métodos por prototipos o las medidas en el sistema real, se facilitan los análisis "what-if" (¿Qué pasaría si ...?) en los que se experimenta con ligeros cambios en el sistema propuesto para intentar mejorar su rendimiento sin invertir en más recursos.

Por último, usando el método de simulación de la herramienta no requerirá conocimientos profundos sobre los modelos matemáticos simples ni una gran experiencia en la estimación de sistemas informáticos distribuidos. La herramienta estará concebida para un uso sencillo, accesible a la mayoría de los usuarios y pretende que pueda ser de ayuda para el aprendizaje de otros métodos de estimación de forma paralela.

1.2. Objetivos

El tutor de este trabajo ha desempeñado el rol de cliente, en el sentido que ha sido quien ha establecido los objetivos y los requisitos que debe cumplir el sistema.

El principal objetivo de este proyecto es crear una aplicación que permita simular la ejecución de un Sistema Informático Distribuido, especialmente sistemas informáticos con múltiples componentes. Entre los sistemas posibles se incluyen:

- Sistemas básicos analizables con modelos matemáticos simples
- Sistemas complejos analizables con modelos matemáticos simples
- Sistemas con retroalimentación
- Sistemas con peticiones con distintos grados de prioridad
- Sistemas con distintas franjas temporales en las entradas de peticiones para permitir picos de trabajo o franjas con baja carga de trabajo.

Estas últimas permitirán simular algunos sistemas que no son analizables mediante modelos matemáticos simples.

La aplicación debe permitir al usuario un fácil acceso a la modificación de los datos y las variables del sistema que está diseñando, así como la funcionalidad de visualizar y poder manipular el sistema de manera gráfica, permitiendo al usuario llevar a cabo análisis "what-if".

También se ha incluido el objetivo de que la ejecución de la simulación se pueda visualizar, tanto de manera gráfica en el propio diagrama como la evolución de los datos para poder detectar con mayor facilidad posibles problemas en el diseño como cuellos de botella.

Con los objetivos mencionados se puede ver que esta aplicación tendrá una doble utilidad:

- Podrá ser usada en el diseño de Sistemas Informáticos Distribuidos para estimar el rendimiento de los diseños propuestos.

- Podrá ser usada para apoyar el aprendizaje de los modelos matemáticos simples, ya que, permite ver de manera gráfica la simulación y se podrán ver como se estabilizan los distintos valores que se estudian en dichos modelos matemáticos simples.

Por último, existen otros dos objetivos que, pese a no ser estrictamente necesarios, serían de gran utilidad:

- Incluir una opción para el cálculo de la media de tiempos de un componente mediante la introducción de un fichero que contenga los datos de tiempos de entrada y salida del componente.

- Que sea una aplicación sea de fácil acceso, que sirva como apoyo pedagógico en el estudio de los modelos matemáticos simples del análisis de sistemas informáticos distribuidos.

1.3. Organización de la memoria

Introducción: La sección actual, donde se presenta el proyecto junto con el problema que pretende resolver y la motivación para empezarlo.

Estado del arte: En esta sección se analizan otras herramientas que solucionan el mismo problema que se pretende resolver, además de las diferentes tecnologías que podrían usarse y se justifica la elección.

Análisis y diseño: En esta sección se detalla el análisis del problema planteado y los requisitos para el proyecto, así como el diseño decidido para cumplir los requisitos.

Desarrollo: En esta sección se detalla la implementación del proyecto y se comenta su desarrollo.

Pruebas y resultados: En esta sección se muestran las pruebas realizadas para validar el proyecto y se comentan sus resultados comparándolos con los obtenidos con modelos matemáticos.

Conclusiones y trabajo futuro: En esta sección se extraen las conclusiones del proyecto observando los límites de este y analizando posibles mejoras

2

Estado del arte

Los Sistemas Informáticos Distribuidos son utilizados habitualmente en la informática actual, por lo que es normal que existan herramientas que intenten dar solución al mismo problema planteado en este trabajo. Como ejemplo, seguidamente se incluyen dos aproximaciones, ya definidas, al problema en cuestión:

• **Queueing Systems Assistance (QSA)** [8]. Esta aplicación se enfoca en el análisis de sistemas simples de un solo servidor. No contempla el análisis de redes de servidores. De los posibles sistemas de un servidor, la aplicación permite analizar 38 tipos de sistema, todos explicados con la notación de Kendall [9]. Esta condición limita el acceso del público en general, dado que requiere conocimientos de los diferentes sistemas en notación Kendall para poder interpretar correctamente los datos obtenidos. Se trata de una aplicación Web y no requiere de ningún tipo de instalación por parte del usuario para utilizarla. Por último, hay que mencionar que usa una nomenclatura distinta a la que se da en la asignatura de Sistemas Informáticos II para algunos de los parámetros devueltos por la herramienta (ver Anexo de pruebas con esta herramienta).

En conclusión, QSA es una herramienta que se aleja de la aproximación elegida en este trabajo respecto al método de análisis, coincidiendo en el método de publicación de la herramienta. En comparación con la aproximación tomada en este trabajo QSA tiene la ventaja de poder analizar más sistemas de un solo servidor y por tanto es más funcional si tenemos que analizar un sistema concreto. En contraste, al no permitir analizar redes de servidores, la herramienta desarrollada en este trabajo toma ventaja dado que dichos análisis si son posibles, además de no ser necesarios tener conocimientos sobre la notación Kendall, que si son requeridos por QSA.

• **Java Modelling Tools (JMT)** [10]. El enfoque de esta aplicación es similar al de este trabajo: una simulación con representación gráfica de la simulación. Nos encontramos que el método de publicación es diferente, ya que se precisa de una instalación de la herramienta. La aplicación requiere de cierto nivel de familiarización para poder conocer todas sus características y funcionalidades. Permite el análisis de redes de servidores, incluyendo algunos componentes adicionales como por ejemplo Joins para juntar las salidas de dos o más componentes en una única entrada, que, si bien pueden complicar el diagrama final del sistema, podrían ayudar a la comprensión de este. La herramienta utiliza diferentes pantallas, la pantalla principal incluye casi exclusivamente la zona para la creación del diagrama del sistema, y ubica en otras ventanas los parámetros de cada componente, las clases de peticiones y los resultados de una simulación. Un punto a destacar es que para obtener los resultados de la simulación debes indicar que resultados se están buscando, en contraste a la herramienta desarrollada en este trabajo que muestra siempre todos los resultados calculados.

En conclusión, JMT es una herramienta más parecida al objetivo de este trabajo, con algunas diferencias. Las ventajas para destacar de JMT frente a la herramienta objeto de este trabajo son la existencia de un apartado gráfico más potente (sobre todo a la hora de mostrar gráficamente los resultados de la simulación), la capacidad de poder incluir más distribuciones y personalizar las estrategias de las colas y otros componentes. En el apartado de desventajas nos encontramos que no permite indicar franjas de tiempo para simular picos de trabajo

o franjas de baja carga de trabajo, ni una función para indicar una salida dentro del sistema para las peticiones rechazadas que si se ofrecen en la herramienta objeto de este trabajo. Por otra parte, existen diferencias respecto a la visualización de la gestión de los datos de los componentes y resultados de la simulación, JMT los distribuye en pantallas diferentes, obligando al usuario a buscar los resultados en otra pantalla, y la herramienta objeto de este trabajo los ofrece en la misma pantalla ofreciendo la posibilidad de ocultarlos o mostrarlos a conveniencia del usuario.

2.1. Estudio de las tecnologías a usar

En esta sección se explicarán las distintas tecnologías que se utilizan en el proyecto, los motivos de su elección y las opciones descartadas.

2.1.1. Plataforma

Dado que se trata de una aplicación en local a la que se accede desde una Web se optó por una plataforma gratuita para ubicarla, Heroku [11], de fácil acceso y que presentaba la conveniencia de que era una plataforma ya conocida por el desarrollador. Las características que apoyaban la elección de esta plataforma eran la no restricción a un único framework o formato de aplicación, ofreciendo la libertad de elegir.

La aplicación, por tanto, está desplegada en Heroku con la URL: https://tfg-production-app.herokuapp.com/ [12]

2.1.2. Framework

La aplicación tiene sentido como SPA (Single Page Application) [13] o aplicación de una sola página, ya que el núcleo de la aplicación será la creación del diagrama, y luego la simulación del mismo, el cual debe mostrar los resultados en tiempo real y de forma gráfica. Por lo tanto, la decisión del framework o formato de la aplicación debe ajustarse a una SPA.

Existen varios frameworks que presentan una buena solución para crear esta aplicación, finalmente se eligió React [14] por los siguientes motivos:

- Modularización: React está organizado en componentes, cada componente necesita una función para renderizar su contenido y puede tener un estado asociado, que serían las variables internas del propio componente, y si el estado cambiase provocaría una renderización. Este encapsulamiento en componentes permite que se puedan instalar componentes ajenos y usarlos sin mayor dificultad. También permite crear componentes propios y encapsular funcionalidad separando las secciones.

- Renderiza solo las partes que se han modificado: React mantiene una representación virtual de lo que renderizaría en pantalla, llamado DOM virtual, el cuál usa para comparar las modificaciones necesarias y solo renderizar las partes que hayan cambiado, de esta forma evita renderizar constantemente la pantalla entera aun cuando solo se haya modificado una sección de la misma.

- Permite combinar HTML con código JavaScript: React tiene un tipo de fichero con extensión ".jsx", en el cual se puede entrelazar código HTML con código JavaScript, evitando tener ficheros separados y facilitando la modificación del HTML final de la página con código. Esto puede tener sus inconveniencias, ya que al tener todos los contenidos de estilo y funcionalidad en un mismo fichero este puede hacerse muy grande y dificultar su comprensión, pero como se ha mencionado antes React facilita la modularización del código.

Se han tenido en cuenta otros frameworks, como por ejemplo Angular [15]. Sin embargo, para el desarrollador React es más familiar, ya que lo ha usado con anterioridad.

2.1.3. Instalador de librerías

Como se ha mencionado en el apartado anterior una de las ventajas de React es que permite instalar componentes y librerías externas con facilidad, por lo que será necesario tener una forma de instalar dichas librerías. React se suele utilizar en combinación con NodeJs [16] que tiene el instalador NPM [17] por defecto.

2.1.4.Diagrama

Para la implementación de este trabajo es necesario mantener en el apartado gráfico un diagrama del sistema y que se pueda modificar de manera gráfica. Esto se sale un poco de la escala del trabajo si fuera necesario implementarlo de cero, pero existen componentes externos que facilitan el manejo de apartados gráficos de ese estilo. Después de encontrar algunos de estos componentes que no alcanzaban la funcionalidad necesaria, o no proporcionaban las facilidades para poder implementarlo se encontró GoJs [18]. Este componente tiene la funcionalidad necesaria y además tiene un componente específico para React, desafortunadamente, se trata de una herramienta de pago. Por ese motivo, finalmente, se eligió Cytoscape.js [19] [20], que pese a no tener un componente específico para React tiene la funcionalidad necesaria y es gratuito.

2.1.5. Distribuciones

La última librería externa necesaria para implementar la simulación debe de permitir la obtención de valores siguiendo una distribución específica, como mínimo se necesitará que tenga las distribuciones de Poisson y Normal. Adicionalmente, y para completar las que se suelen usar en el ámbito de los modelos matemáticos simples, convendría que tuviera las distribuciones de Gamma, General y Erlang. Random [21] cumple con las distribuciones requeridas y además tiene otras que están en desarrollo. Con esa librería cumplimos con lo estrictamente necesario, pero no tiene ninguna de las distribuciones adicionales. De las distribuciones adicionales, solo se encontró una Gamma, para la que se usará la librería D3 [22].

2.1.6. Estilos

Por último, con objeto de mejorar la parte gráfica de la aplicación se utilizan varias librerías que facilitan el uso de componentes con estilos propios o iconos:

- Fontawesome [23] es una conocida librería de imágenes e iconos gratuita que, en este caso, se utiliza para los iconos de la aplicación.

- Reactstrap [24] es una versión adaptada a React de Bootstrap [25] otra conocida librería de estilos para HTML y que, para la herramienta objeto de este trabajo, se utiliza en los formularios de introducción de datos y las alertas de la aplicación.

3

Diseño

3.1. Requisitos

A continuación, se inicia la descripción del análisis y el diseño con los requisitos funcionales y no funcionales de la aplicación propuesta.

Dentro de los Sistemas Informáticos Distribuidos que vamos a tratar la velocidad de entrada y salida de peticiones de un componente se puede medir en peticiones por unidad de tiempo o unidades de tiempo entre peticiones. Para abreviar, ya que se menciona varias veces las tasas referidas serán siempre en unidades de tiempo entre peticiones.

3.1.1. Requisitos funcionales

RF-1. La aplicación debe tener al menos un método gráfico para crear el sistema a simular.

RF-2. La aplicación debe poder simular el sistema mostrando de manera gráfica su evolución.

RF-3. La aplicación debe mostrar los cambios que ocurran durante la simulación.

RF-4. La aplicación debe permitir distinguir entre mínimo tres distribuciones distintas para las tasas de los componentes:

- Poisson.
- Normal.
- Constante.

RF-5. Adicionalmente la aplicación debe permitir la fácil introducción de distribuciones para las tasas de los componentes

RF-6. La aplicación debe permitir añadir componentes al sistema, los cuales son:

- Input: Un componente que permite la entrada de peticiones al sistema.
- Servidor: Un componente que permite el procesado de peticiones.
- Balanceador: Un componente que permite la distribución de peticiones a varios componentes.

- Output: Un componente que permite la salida de las peticiones del sistema.

RF-7. La aplicación debe guardar los parámetros necesarios para la simulación de cada componente, los cuales son:

- Input
    - o Nombre
    - o Tasa de entrada al sistema
    - o Distribución de la tasa de entrada al sistema
    - o Prioridad de las peticiones generadas
    - o Duración del periodo activo del input, en caso de que le imponga una franja de activación y desactivación
    - o Duración del periodo inactivo del input, en caso de que le imponga una franja de activación y desactivación
- Servidor
    - o Nombre
    - o Lista de procesadores, cada uno con los siguientes parámetros:
        - Tasa de salida
        - Distribución de la tasa de salida
    - o Tamaño de la cola
- Balanceador
    - o Nombre
    - o Método de elección del destino de cada petición:
        - Por probabilidad, que permitirá especificar la probabilidad de que una petición sea redirigida a una rama específica.
        - Por opción, que buscará la primera rama que acepte la petición.
- Output
    - o Nombre

RF-8. La aplicación debe permitir conectar los componentes entre sí para marcar las entradas y salidas de cada componente.

RF-9. La aplicación debe calcular los siguientes datos durante la simulación, los cuales son:

- Input
    - o Peticiones creadas
- Servidor
    - o Tasa de entrada medida
    - o Peticiones encoladas

    - Tasa de salida medida
    - Peticiones procesadas
    - Estado de la cola
    - Máximo alcanzado por la cola
    - Peticiones rechazadas
    - Peticiones recibidas

- Balanceador
    - Peticiones redirigidas

- Output
    - Peticiones terminadas

RF-10. La aplicación debe mostrar los siguientes datos calculados al final de la simulación:

- Globales
    - L calculada en el sistema

- Input
    - Nombre
    - Tasa de producción medida
    - Total de peticiones producidas

- Servidor
    - Nombre
    - Número de procesadores
    - Total de peticiones recibidas
    - Total de peticiones procesadas
    - Máximo alcanzado por la cola
    - Rho media
    - L media
    - Lq media
    - W media
    - Wq media
    - Lq / Wq media

    - Por cada procesador
        - Tasa de llegada medida
        - Tasa de salida medida
        - Lambda
        - Mu
        - Tiempo procesando
        - Tiempo ocioso
        - Tiempo total
        - Rho medida en el sistema
        - Peticiones procesadas
- Balanceador
    - Nombre
    - Método de elección de destino de las peticiones
    - Tasa de llegada medida
    - Total de peticiones redirigidas
- Output
    - Nombre
    - W medio

RF-11. La aplicación debe permitir modificaciones en los datos de los componentes mientras no esté simulando.

RF-12. La aplicación debe permitir definir las unidades de tiempo de duración de la simulación.

RF-13. La aplicación debe permitir definir las unidades de tiempo de “ramp-up” de la simulación, para permitir que los parámetros de esta se estabilicen antes de empezar los cálculos.

RF-14. La aplicación debe ser capaz de prever errores simples en el sistema dado antes de comenzar la simulación, como, por ejemplo, parámetros necesarios no especificados o falta de un componente de entrada o salida para las peticiones al sistema.

RF-15. La aplicación debe permitir el acceso y la modificación de los parámetros necesarios para la simulación específicos de cada componente, mientras no se esté simulando.

RF-16. La aplicación debe permitir el acceso a los parámetros necesarios para la simulación y los datos obtenidos durante la simulación específicos de cada componente mientras se está simulando.

RF-17. La aplicación debe permitir que el usuario introduzca un fichero con los tiempos de un componente para calcular la tasa media de dicho componente.

RF-18. La aplicación debe permitir guardar el sistema creado para en un futuro volverlo a usar.

RF-19. La aplicación debe permitir indicar un número de prioridad por cada input permitiendo indicar que ciertas peticiones tienen prioridad sobre otras en la cola de un servidor durante la simulación.

RF-20. La aplicación debe permitir indicar unas franjas de tiempo en las cuales pueda haber picos de trabajo o franjas de tiempo con menor carga de trabajo.

3.1.2. Requisitos no funcionales

RNF-1. Interfaz amigable con el usuario y que facilite las pruebas “what-if”.

RNF-2. Fácil acceso para la mayoría de los usuarios y compatibilidad con los sistemas más comunes.

RNF-3. El rendimiento de la aplicación debe ser razonable, sobre todo durante la simulación en con apartado gráfico.

3.2. Patrón de diseño

Se distinguen dos partes en esta aplicación, la interfaz que usará el usuario para interactuar con el sistema que este creando, y el programa que simula el sistema creado por el usuario. Esta división marca un patrón a seguir bastante claro, el patrón de Modelo-Vista-Controlador [26].

El patrón de Modelo-Vista-Controlador separa una aplicación en tres partes, la vista, el modelo y el controlador:

- La vista sería la interfaz que se encargaría de recibir y procesar los inputs del usuario.
- El modelo sería el programa interno que manipula los datos de la aplicación y ejecuta los procesos de esta.
- El controlador es el puente comunicador entre ambas partes, permitiendo que la vista no tenga que adaptarse al modelo y viceversa. En los siguientes apartados se incluyen más comentarios acerca del diseño de cada parte.

3.3. Arquitectura

Según los requisitos de la aplicación debe haber una interfaz gráfica para crear y modificar el sistema a simular, esto junto con los requisitos no funcionales de compatibilidad y fácil acceso orientan a una aplicación que no dependa exclusivamente del entorno en el que se ejecuta. Por ello se han tenido en cuenta dos opciones:

- Implementar la aplicación en un lenguaje que no dependa del entorno, por ejemplo, Java.
- Crear una aplicación Web de manera que se usará un navegador para acceder a la aplicación.

La elección recae en la segunda opción, es decir, se crea una aplicación Web para que la aplicación se ejecute en el navegador, que no requiere ningún tipo de instalación y facilita el acceso a la misma.

Dentro de las aplicaciones Web existe un patrón que combina bastante bien con el tipo de aplicación que se pretende crear en este proyecto: SPA. Las SPA (Single Page Application) son aplicaciones que tienen solamente una

URL y, por tanto, un único HTML. Las SPA sólo deben cargarse una vez, o los recursos se cargan dinámicamente cuando se necesiten. Este tipo de aplicaciones tienen el objetivo de dar una experiencia fluida. Vistos los objetivos de esta aplicación se puede ver cómo gran parte de esta aplicación debe estar en una sola página.

Se podría dividir la experiencia del usuario en dos partes:

- Creación del sistema mediante la introducción de los datos de cada componente y los parámetros de la ejecución.

- Ejecución de la simulación y muestra de los resultados.

La única sección gráfica diferente entre estas dos partes sería la sección donde se muestra el resultado total de la ejecución, pero tanto la información de los parámetros de la ejecución, las unidades de tiempo transcurridas o la información específica de cada componente, se pueden mostrar en secciones ya existentes. En estos casos, las unidades de tiempo transcurridas se pueden mostrar en la sección donde se especifican las unidades de tiempo que debe durar la simulación y la información específica de cada componente se puede mostrar donde se introducen los parámetros específicos de cada componente. Además, el propio diagrama del sistema se puede usar para mostrar gráficamente la evolución de la simulación.

Por estos motivos, se puede ver que montar la aplicación como una SPA tiene varias ventajas. Además de la fluidez en la experiencia del usuario, si la información no debe ser cargada más de una vez no hay que preocuparse de transferirla, ya que se mantiene durante todo el uso de la aplicación.

Por otro lado, tampoco habría que preocuparse de borrar la información, ya que al no usar ningún almacenamiento permanente en el navegador como caché o cookies, con el simple hecho de recargar la página podemos reiniciar la aplicación sin mayor problema.

Como ya se explicó en la sección Estudio de tecnologías a usar el framework elegido para esta aplicación es React. El concepto de componente de React permite margen suficiente para modularizar la página, dado que un componente puede tener toda la lógica que se necesite, mientras siga unas reglas de herencia simple. Como por ejemplo tener una función render que devuelva el HTML resultante del componente. Luego ese componente se usará como una etiqueta HTML normal. Por lo tanto, los componentes son útiles principalmente para las partes que, o bien tienen una lógica a parte del resto, por ejemplo, el formulario de introducción de datos de cada componente del sistema o el propio sistema; o bien son un componente que se va a usar varias veces, lo cual facilitará la reutilización del código.

Por último, una aplicación Web tiene dos partes, la parte del cliente y la parte del servidor. Para este trabajo se decidió que toda la lógica de la aplicación debía estar ubicada en la parte del cliente, dejando el servidor como la fuente desde la que obtener el programa al cargar la página. Otra opción que se tuvo en cuenta fue la de mantener el modelo, es decir, el programa que ejecuta la simulación, en la parte del servidor, pero esta opción planteaba un problema: la comunicación del estado de la simulación durante la simulación en tiempo real. Si se mantuviera la conexión con el servidor abierta durante toda la simulación podríamos continuar la comunicación del estado durante toda la simulación, pero uno de los objetivos de la aplicación es que se pueda usar de manera pedagógica como apoyo en el aprendizaje de los modelos matemáticos simples. Siguiendo ese objetivo, la aplicación se podría utilizar en una clase para hacer un ejercicio y, en ese caso, habría que tener en cuenta el número máximo de conexiones que podría hacer el servidor en el que se aloje la aplicación.

Por estos motivos se decidió mantener la aplicación en la parte del cliente, dado que, así, se permite una mejor escalabilidad y se evitan problemas de comunicación entre modelo y vista.

3.4. Modelo

En esta sección se explicará el diseño del modelo de la aplicación, es decir, del programa que simula el sistema creado en la vista de la aplicación. Primeramente, hay que mencionar que, como indica el patrón MVC, todo el programa simulador está completamente separado de la vista de la aplicación, de manera que en ningún momento la vista puede interferir de ninguna manera con la simulación. Además, y para evitar otros posibles problemas de coordinación, se decidió que la simulación fuera una simulación de eventos discretos. De esta manera, aunque existe un parámetro de tiempo en la simulación no existe un reloj que avance con el tiempo, sino que son los eventos los que hacen avanzar el reloj hasta el momento en el que ocurren cuando se ejecutan. Además, con este tipo de simulación fácilmente el controlador puede intercalar las renderizaciones de la vista, ya que la simulación se hace por bloques, siendo cada bloque un evento. Adicionalmente, al no haber un reloj que avance con el tiempo las unidades de tiempo quedan a disposición de lo que seleccione el propio usuario.

Para comenzar a hablar del diseño del programa simulador se debe hablar de sus partes, aunque antes de nada hay que hacer una aclaración, en JavaScript existen las clases y la herencia, pero no es tan necesaria como podría ser en Java, ya que no hace falta indicar el tipo de dato al tratarlo. De esta manera, aunque no exista una clase Componente de la que hereden las clases de los inputs o los servidores, con el fin de simplificar los diagramas y las explicaciones se trataran todos como si heredaran de la clase Componente. En las siguientes figuras se pueden ver los diagramas de clases del modelo, en el que existen objetos que no aparecen explicados como Reporte o Media, estos se tratan de objetos que contienen todos los parámetros calculados por el Componente y los datos de la Tasa del Componente respectivamente.

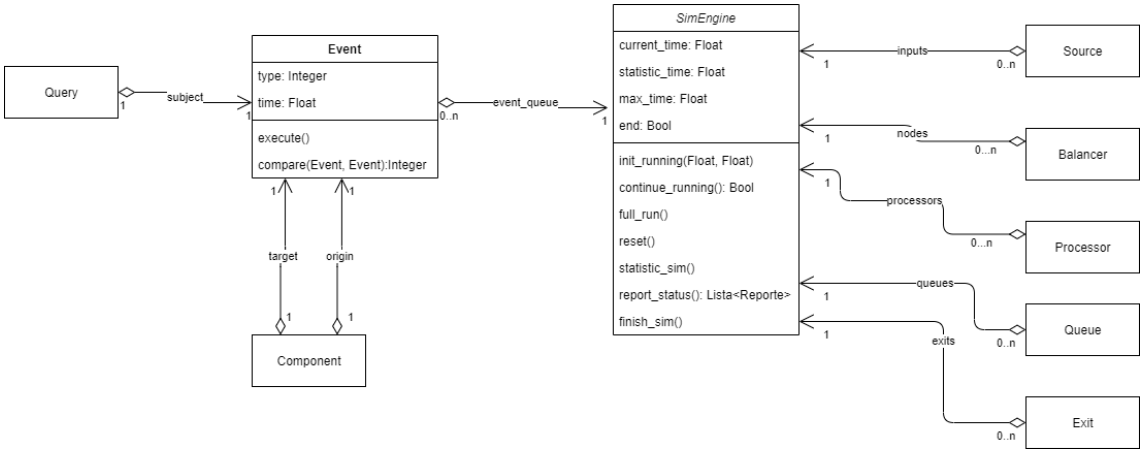

Figura 1.1: Diagrama de clase de SimEngine

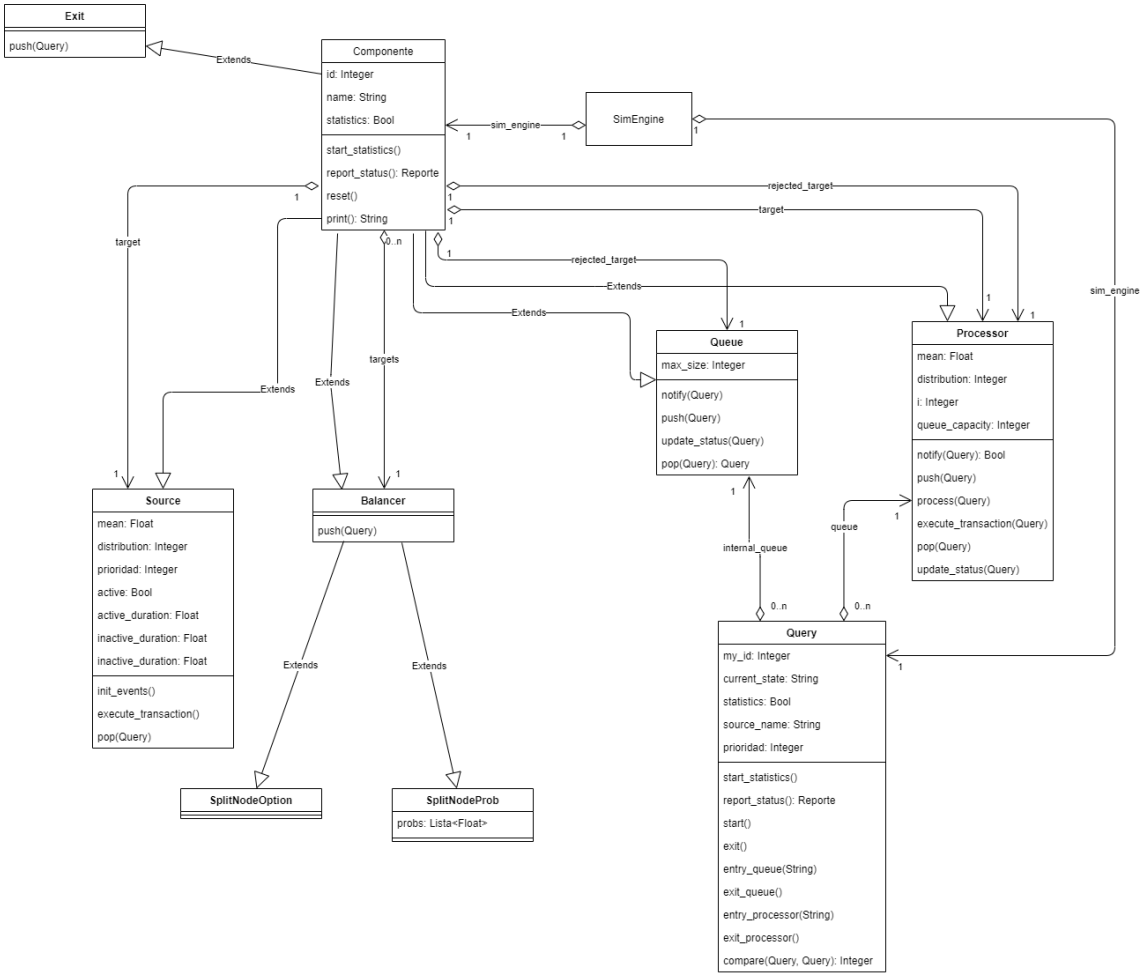

Figura 1.2: Diagrama de clase de Componente

Una vez hecha la aclaración, primero se explicará SimEngine. Esta parte contiene toda la información de la simulación, incluidos todos los Componentes del sistema a simular y la cola de eventos de la simulación, también maneja el reloj de la simulación. Dentro de operaciones sólo hay que destacar las dos modalidades de simulación, full_run() que ejecuta el bucle entero de simulación hasta que llegue al tiempo máximo, y continue_running() que ejecuta un único evento de la cola y devuelve si la simulación continua después de ese evento.

Cada Componente mantiene su propia información, aparte de la información común como el id o el nombre del Componente o las funciones report_status() para reportar el estado al finalizar la simulación. Además, los Componentes que pueden ejecutar eventos, Processor y Source tienen las funciones de execute_transition(Query) y pop(Query) para ejecutar la petición de un evento o para extraer la petición y poder pasarla al siguiente Componente. Es importante comentar que Processor es la clase de un único procesador con una cola, para servidores con más de un procesador se añade el componente Queue que se asocia con tantos Processor como tenga el servidor para unificar la cola del servidor, pasando el Queue a ser la conexión con todo Componente que envíe Querys al servidor y este las reparte entre los Processor según estén libres.

Continuamos hablando del Balancer el cual ofrece dos variantes: por opción o por probabilidad. El Balancer por opción distribuye las peticiones entrantes intentando pasárselas a los Componentes en el orden en el que se hayan introducido los Componentes como destinos en este Balancer, de manera que intentará pasársela al segundo Componente si el primero rechaza la petición.

El Balancer por probabilidad tiene asociado una probabilidad a cada Componente destino, la cual requiere una

introducción del dato por parte del usuario, de esta manera se calculará un número aleatorio y con este número se elegirá el Componente al que pasar la petición.

Por último, los objetos Event y Query mantienen la información de un evento en la simulación y una petición respectivamente. El primero contiene la Query afectada por el evento y el Componente objetivo del evento, en el que se ejecutará. El segundo mantiene la información de la petición calculando los tiempos de espera y ejecución mientras esta en el sistema.

La ejecución de un evento en la simulación es simple, solo requiere obtener el siguiente evento en orden cronológico, avanzar el reloj hasta el momento del evento y ejecutarlo si no se ha sobrepasado el tiempo máximo (véase la figura 1.3). Las ejecuciones de eventos son algo más complejas, porque deben calcular los parámetros en cada movimiento para poder mostrarlos al final de la simulación. En esto, los componentes Processor y Queue son los más complejos, ya que intervienen en el cálculo de una mayor cantidad de parámetros.

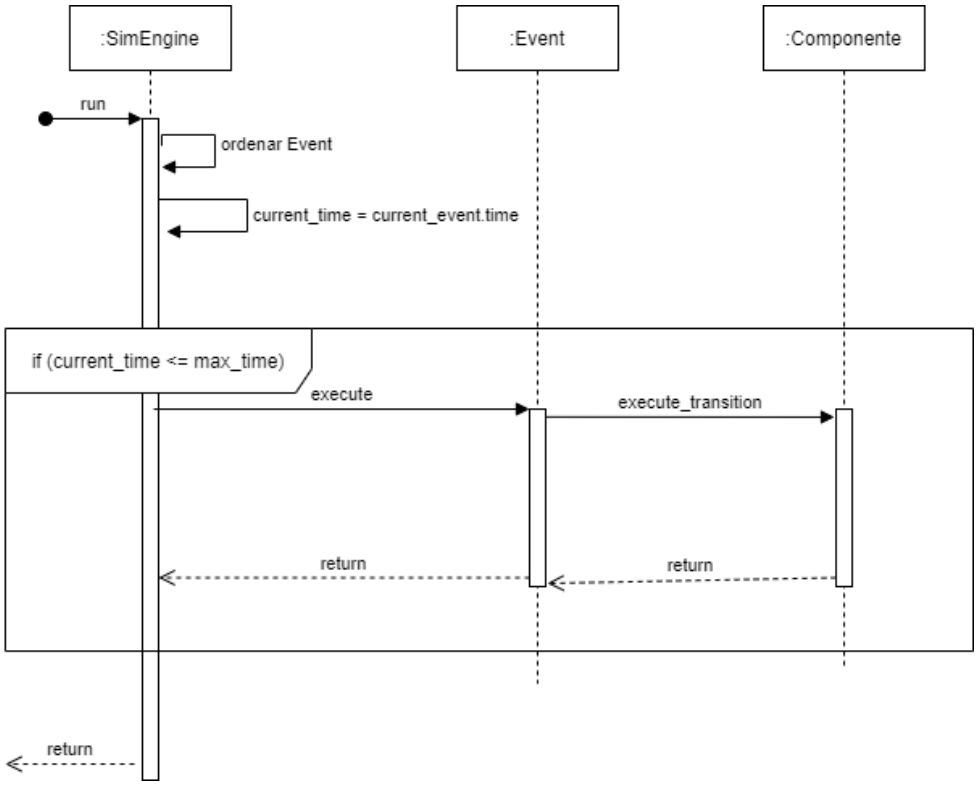

Figura 1.3: Diagrama de secuencia de la ejecución de la simulación

3.5. Controlador

La funcionalidad del Controlador consiste en mantener el diagrama como un grafo de Nodos con la información que introduce el usuario y permite crear los Componentes a partir de esos Nodos. Adicionalmente, permite controlar la simulación, iniciándola o parándola. Por último, permite validar que el diagrama pasado como grafo de Nodos es correcto para crear una simulación y modificar las imágenes del diagrama mostrado en caso de que un input entre en una franja inactiva o que un servidor rechace alguna petición.

3.6. Vista

Para la vista es necesario tener en cuenta los tres estados en los que puede estar la aplicación:

Inicial: En este estado puede haber un sistema en construcción, pero no se ha ejecutado ninguna simulación. Se pueden manipular los parámetros necesarios para la simulación de cada componente, pero no al no haberse simulado el sistema no tiene ningún dato que se genera durante la simulación y por lo tanto no se muestran.

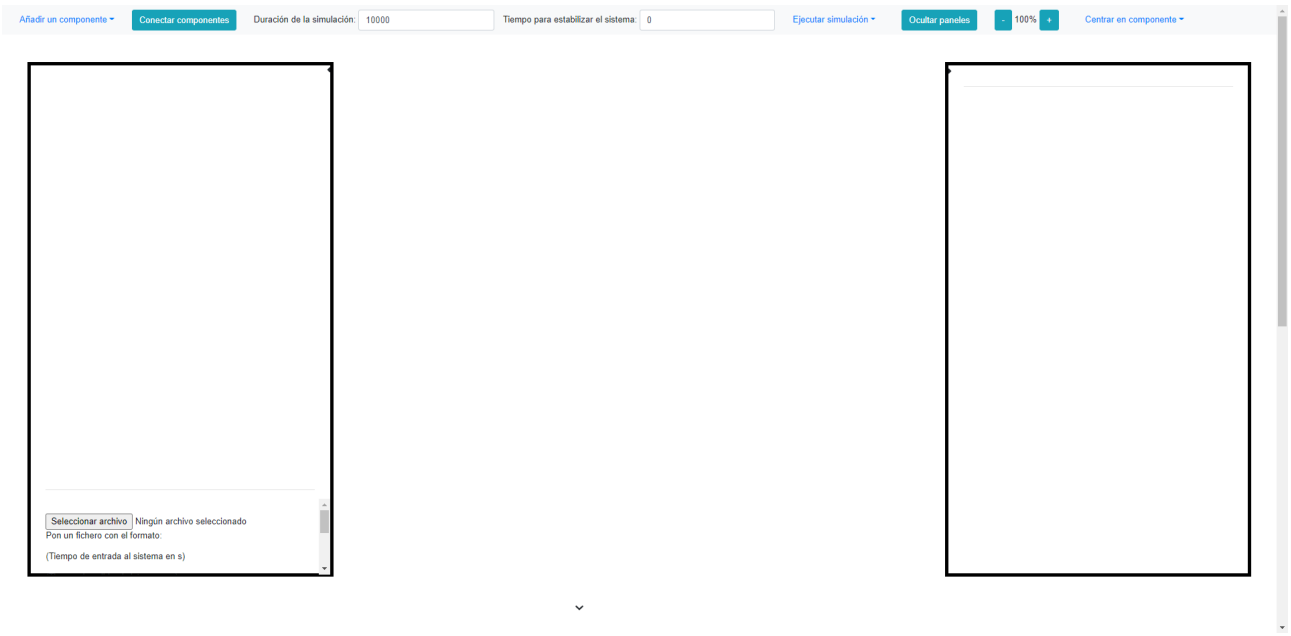

Figura 2.1: Estado inicial de la aplicación

Simulando: En este estado se está simulando el sistema construido. Se pueden ver los parámetros necesarios para la simulación de cada componente y se pueden ver también los datos generados en la simulación, pero no se puede modificar nada del sistema.

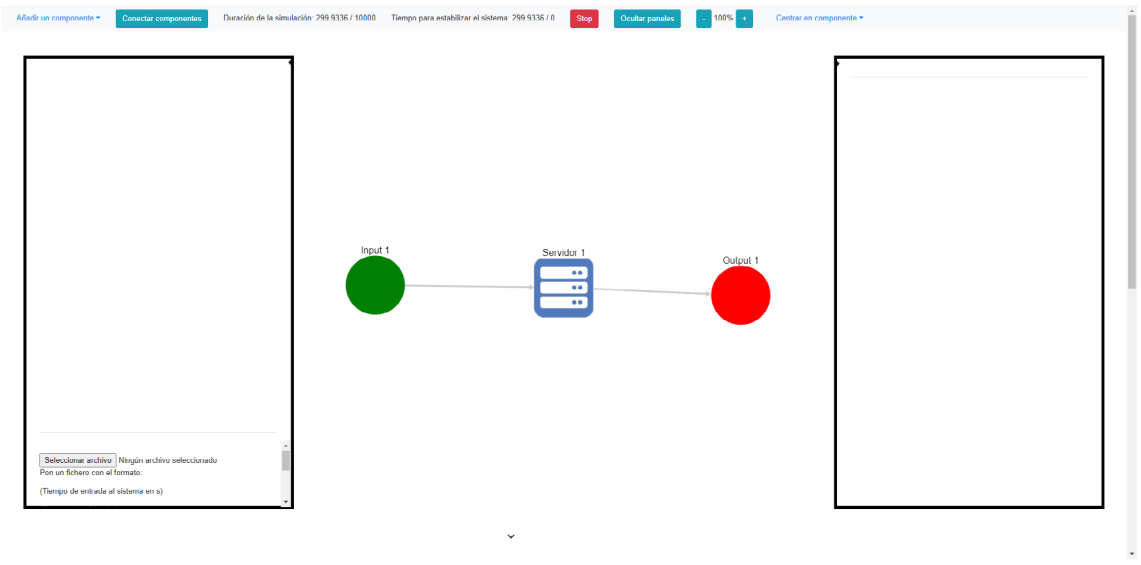

Figura 2.2: Estado simulando de la aplicación

Resultados: En este estado se ha terminado una simulación y se pueden ver sus resultados, el sistema vuelve a ser modificable y además se pueden ver los datos generados en la anterior simulación de cada componente.

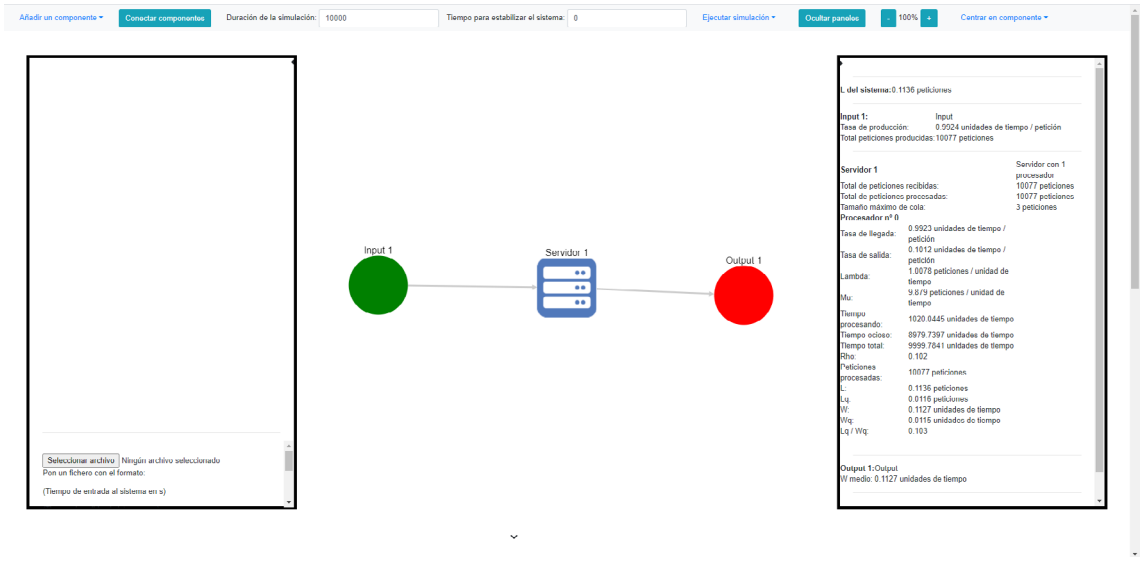

Figura 2.3 Estado resultados de la aplicación

También podemos distinguir varias secciones que deben tener la interfaz, estas secciones son:

- Menú: En esta sección se puede manejar casi todos los apartados de la aplicación. Estos apartados son, de izquierda a derecha:

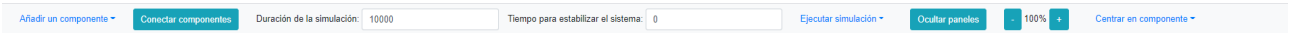

Figura 2.4 Sección menú

o Añadir nuevos componentes: Es un dropdown para añadir cualquier tipo de componente al diagrama, el diagrama se centrará automáticamente en el nuevo componente.

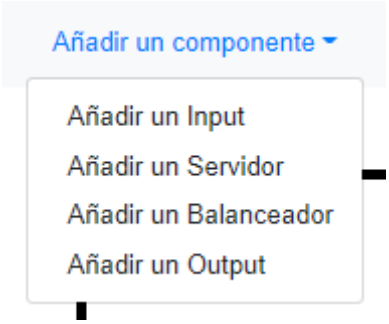

Figura 2.5 Apartado añadir nuevo componente

o Conectar componentes: Este apartado tiene 3 posibles estados:

El inicial, en el que todavía no se ha comenzado a conectar.

El segundo, en el que se debe seleccionar el primer componente a conectar, a partir de este estado aparece un botón para cancelar la conexión de componentes.

El tercero, en el que se debe seleccionar el segundo componente a conectar.

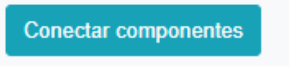

Figura 2.6 Apartado conectar componentes estado 1

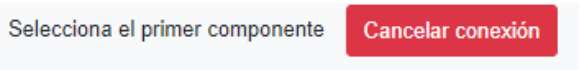

Figura 2.7 Apartado conectar componentes estado 2

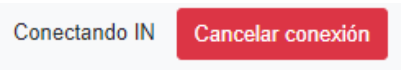

Figura 2.8 Apartado conectar componentes estado 3

o Duración de la simulación: En este apartado se puede indicar la duración de la próxima simulación en unidades de tiempo de simulación, además de la duración del periodo de ramp-up

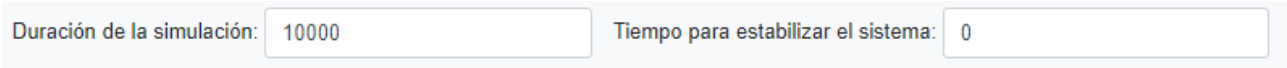

Figura 2.9 Apartado duración de la simulación

o Ejecución de simulación: En este apartado se puede comenzar la simulación, en cualquiera de sus dos modalidades: Ejecución rápida, sin apartado gráfico, o Ejecución gráfica, que muestra en cómo va avanzando la simulación por pantalla.

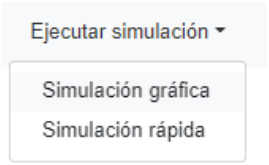

Figura 2.10 Apartado ejecución de simulación

o Ocultar paneles: Este botón te permite ocultar los paneles laterales para tener más espacio para manipular el diagrama.

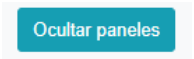

Figura 2.11 Apartado ocultar paneles

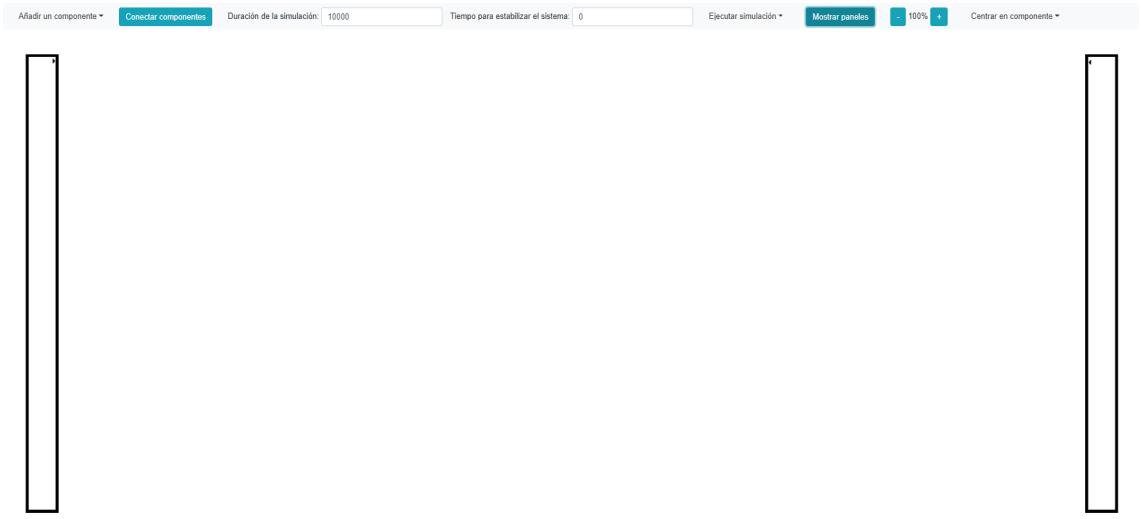

Figura 2.12 Paneles ocultos

o Manejo de Zoom y Centrar: En este apartado se puede modificar el zoom y la posición del diagrama. Con el dropdown de Centrar en componente se puede elegir uno de los componentes en el grafo y se centrara la pantalla en él.

Figura 2.13 Apartado Zoom y Centrar

- Sección de los datos específicos: En esta sección se mostrarán los datos específicos del componente seleccionado, también se podrán modificar los parámetros cuando sea posible. Para que aparezcan los datos específicos de un componente hay que hacer click en él. Esta sección puede minimizarse para dar espacio a otras secciones como la del sistema haciendo click en el triángulo en la esquina derecha en la parte superior.

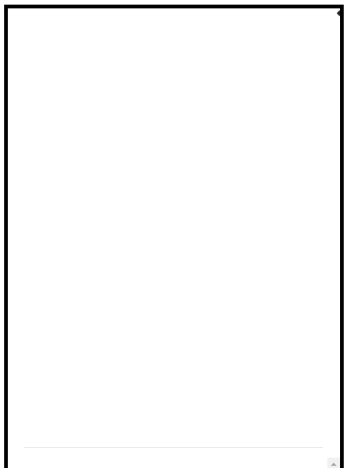

Figura 2.14 Sección de datos específicos

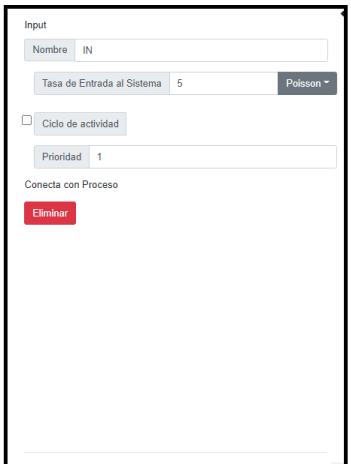

Figura 2.15 Sección de datos específicos de input

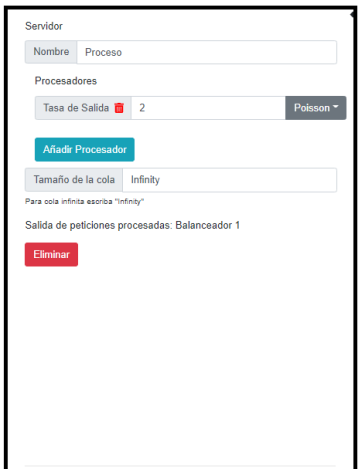

Figura 2.16 Sección de datos específicos de servidor

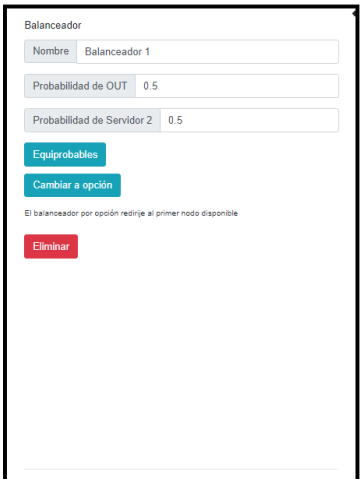

Figura 2.17 Sección de datos específicos de balanceador

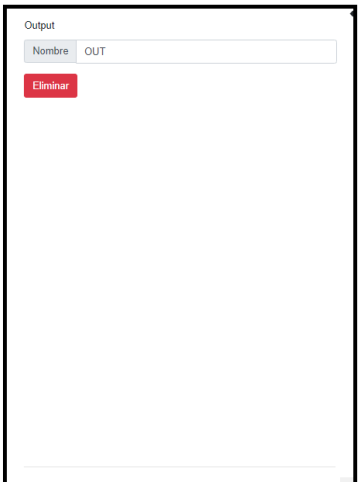

Figura 2.18 Sección de datos específicos de output

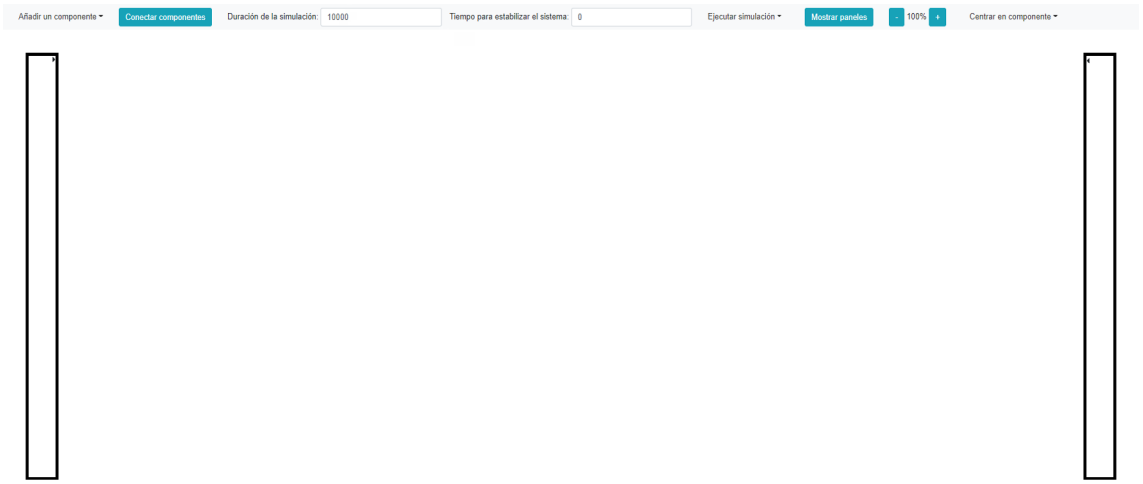

Figura 2.19 Sección de datos específicos oculta

- Sección del diagrama: En esta sección se podrá ver y modificar de manera gráfica el sistema, además de poder seleccionar el componente que quiera el usuario para ver sus datos en la sección anterior.

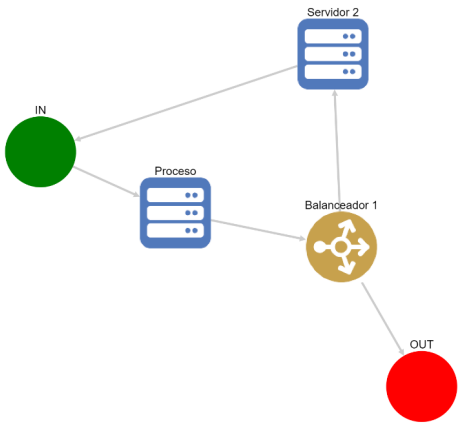

Figura 2.20 Sección del diagrama

- Sección de resultados de simulación: Esta sección servirá para mostrar los resultados de una simulación finalizada o los errores al intentar simular un sistema incorrecto. Esta sección no es necesaria en todos los estados de la aplicación así que podría minimizarse para dar espacio a otras secciones como la del sistema haciendo click en el triángulo en la esquina izquierda en la parte superior.

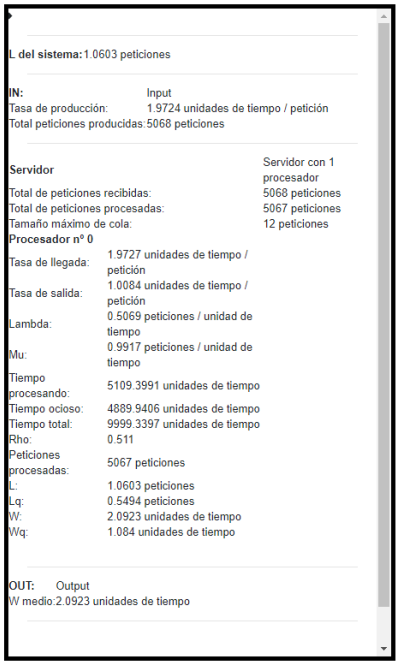

Figura 2.21 Sección de resultados de la simulación

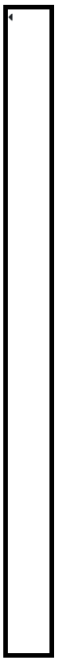

Figura 2.22 Sección de resultados de la simulación oculta

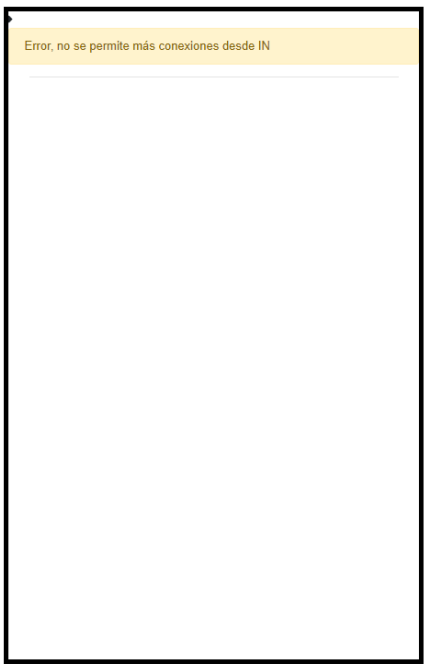

Figura 2.23 Sección de resultados de la simulación con errores

- Sección de cálculo de medias: En esta sección únicamente se explica el formato que debe tener el fichero a introducir y se podrá introducir el fichero, una vez se haya introducido un fichero válido se mostrará también la media resultante.

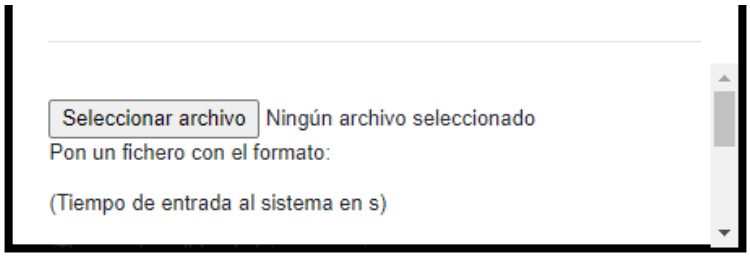

Figura 2.24 Sección de cálculo de medias

- Sección de guardado del diagrama: En esta sección se podrá poner un Json con la información de un sistema para que la aplicación lo utilice, de la misma manera se podrá obtener el Json del sistema que actualmente se está creando para reusarlo en un futuro. Esta sección no es importante durante el uso de la aplicación, cuando más se usará será al terminar con un sistema creado o al empezar con un sistema previamente creado, por lo que puede estar apartada de la vista, pero debe estar señalizada su existencia para no ser pasada por alto.

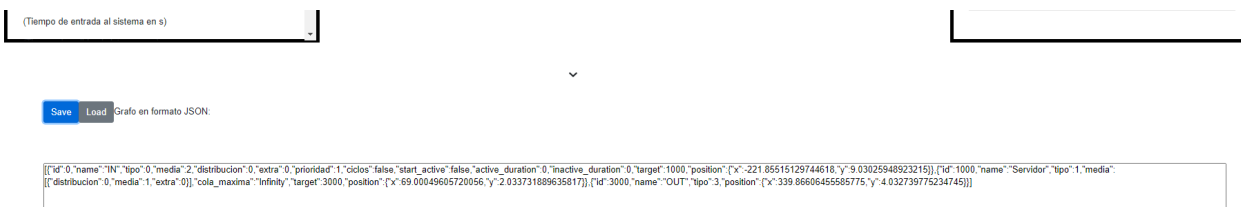

Figura 2.25 Sección de guardado del diagrama

Por último, se mostrará cómo se indica gráficamente el estado de los componentes durante la simulación.

Solo hay dos componentes que pueden cambiar de estado gráficamente.

- Los inputs, que cambian de color de verde a naranja cuando pasan a estar inactivos.

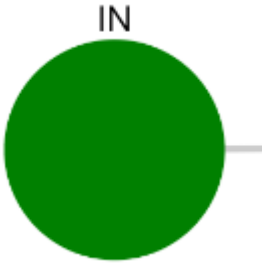

Figura 2.26 Input activo

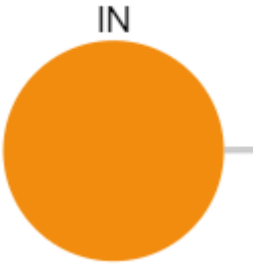

Figura 2.27 Input inactivo

- Los servidores que tienen tres estados dependiendo de las tasas de entrada y salida en el servidor y si ha rechazado peticiones. Un servidor con una tasa de salida menor a la tasa de entrada tendrá una cola estable y por tanto se mostrará de color azul. En el caso de que tuviera una tasa de entrada menor a su tasa de salida, la cola estaría llenándose y se mostrará de color naranja. Y si el servidor ha rechazado alguna petición se mostrará de color rojo.

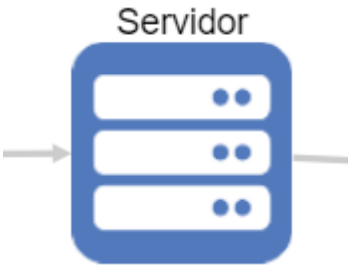

Figura 2.28 Servidor con cola estable

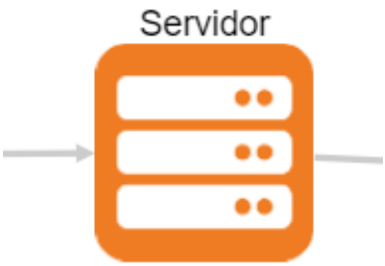

Figura 2.29 Servidor con cola llenándose

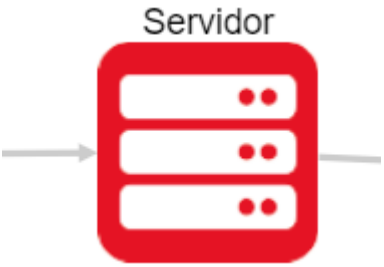

Figura 2.30 Servidor con peticiones rechazadas

4

Desarrollo

En esta sección se explica el funcionamiento del programa simulador y los problemas encontrados durante el desarrollo de la aplicación, así como las soluciones aplicadas.

4.1. Simulación

En este trabajo se usa la simulación, en concreto simulación de eventos discretos, con el objetivo de evitar las simplificaciones que se deben hacer para poder aplicar los modelos matemáticos simples. Debido a esto más allá de comparar con los resultados de algún ejercicio teórico no se incluirán explicaciones sobre el funcionamiento de estos modelos matemáticos simples.

A continuación, se va a explicar los distintos parámetros que se obtienen durante la simulación y el método de generación de los mismos:

Globales

- L calculada en el sistema: Es la suma de las L de cada procesador dentro del sistema.

Input

- Tasa de producción medida: Es la media calculada del tiempo entre entradas de peticiones al sistema por este Input.
- Total de peticiones producidas

Servidor

- Total de peticiones recibidas: Total de peticiones recibidas por el servidor.
- Total de peticiones procesadas
- Tamaño máximo de cola alcanzado
- En caso de tener más de un procesador:
  - Rho del servidor: Es la rho calculada, o intensidad de tráfico efectiva.
  - L del servidor: Es el número medio de clientes en el servidor
  - Lq del servidor: Es el número medio de clientes en la cola del servidor
  - W del servidor: Es el tiempo medio de respuesta del servidor

- Wq del servidor: Es el tiempo medio de espera en la cola del servidor

o Por cada procesador

- Tasa de llegada medida: Es la media calculada del tiempo entre entradas de peticiones al procesador.

- Tasa de salida medida: Es la media calculada del tiempo entre salidas de peticiones del procesador.

- Lambda: Es el número medio de llegadas por unidad de tiempo, se calcula dividiendo 1 entre Tasa de llegadas medida.

- Mu: Es el número medio de salidas por unidad de tiempo, se calcula dividiendo 1 entre Tasa de salida medida.

- Tiempo procesando: Es el sumatorio de los tiempos de procesamiento del procesador.

- Tiempo ocioso: Es el sumatorio de los tiempos entre procesamientos.

- Tiempo total: Es el tiempo total que ha estado activo, se calcula restando el tiempo de inicio de los cálculos de la simulación al tiempo final de la simulación.

- Peticiones procesadas: Es el total de peticiones procesadas por el procesador.

- En caso de tener un único procesador:

+ Rho: Es la rho calculada, o intensidad de tráfico efectiva.

+ L: Este es el número de peticiones medio en el servidor.

+ Lq: Este es el número de peticiones media en la cola del servidor.

+ W: Este es el tiempo medio de estancia en el servidor.

+ Wq: Este es el tiempo medio de espera en cola del servidor.

Balanceador

o Tasa de llegada medida: Es la media calculada del tiempo entre entradas de peticiones al balanceador.

o Total de peticiones redirigidas

Output

o W medio: Es el tiempo medio de las peticiones en el sistema, calculando el tiempo en el que salieron menos el tiempo en el que entraron.

Para crear el programa que simulará los tiempos de procesamiento y espera de un servidor o red de servidores se optó por crear el programa primero en Python, ya que se parece a JavaScript y era más familiar para el desarrollador, de esta manera se obtendría una prueba de funcionamiento antes de empezar con la aplicación final.

Una vez comprobado el funcionamiento del programa que simularía la red de servidores había que pasarlo a JavaScript y crear la aplicación que lo iba a utilizar, optando por una SPA creada con React.

Al comenzar la aplicación en React, ya se tenía una app funcional desde un principio, dado que la herramienta “create-react-app” [27] crea una app básica con un icono que puedes desplegar en local. A partir de esta app básica se podían ir añadiendo tanto nuevos componentes gráficos como nueva funcionalidad y probarla al momento desplegándose en local. Además, NPM permite que al guardar un cambio en un fichero de la app mientras está desplegada se recompile, evitando así tener que replegar y desplegar de nuevo la app cada vez que se añade un cambio, haciendo posible un desarrollo más rápido. React también tiene facilidades a la hora de probar y en el proceso de debug, además de

que los cambios se aplican al momento de guardar hay un plugin de React para Google Chrome [28] que te permite acceder al estado y las propiedades (llamados props) de cada componente durante la ejecución.

4.2. Problemas encontrados durante el desarrollo

A la hora de incluir la creación del diagrama en la aplicación aparecieron algunos problemas de comprensión de la documentación obtenida de Cytoscape. Cytoscape no tiene una forma gráfica para que el usuario interactúe con el diagrama creado, más allá de moverse por el espacio disponible y mover los componentes del diagrama. Por ejemplo, no existen secciones en los componentes que permitan con un click conectar dos componentes como si ofrecen otras aplicaciones de diagramas. Aunque sí permite manipular el grafo de una forma muy sencilla a través de funciones, por ello se añadió la sección para añadir componentes y la sección para conectar componentes al Menú para poder acceder a esta funcionalidad del diagrama. Sin embargo, como en el momento de crear el componente no se ha indicado la posición donde debe crearse, al estar el botón para crearlo fuera, el nuevo componente se crea en la posición inicial y se centra el diagrama en dicho nuevo componente.

Los mayores problemas que surgidos durante el desarrollo de este TFG han sido problemas asociados con la creación del apartado gráfico con HTML y React, debido a que el desarrollador no estaba muy familiarizado con los lenguajes y las formas de ejecución, por este motivo, por ejemplo, no se han conseguido los resultados con la calidad inicialmente planteado, suponiendo un reto y una gran experiencia para el desarrollador.

Otro problema encontrado fue la ejecución de la simulación mostrando la evolución de la simulación, ya que en este caso una ejecución continuada de la simulación impedía a la vista renderizarse porque seguía modificándose el estado de los componentes afectados. Este problema se debió a un fallo en el entendimiento del ciclo natural de actualización de React, ya que espera a la finalización de cualquier función dentro de los componentes antes de renderizar, para evitar modificaciones con cambios de estado parciales. Por lo que una ejecución continuada iba a renderizarse una única vez, al final de la simulación. La solución implementada consistió en sacar el bucle fuera de la simulación y dejarlo en el controlador, de manera que este ejecutara un evento de la simulación permitiendo que se renderice la vista pausando momentáneamente la simulación.

Otra alternativa a la solución mencionada era paralelizar la simulación y la renderización de la vista, utilizando JavaScript, pese a que existen los Workers [29], no existe memoria compartida, por lo que habría que transferir los datos del diagrama. Se descartó esta alternativa dado que la ejecución de un único evento no es un proceso de una carga excesiva y habría que haber mantenido la comunicación cada vez que se ejecutaba un evento.

Adicionalmente se implementó la versión de la ejecución completa sin apartado gráfico, ya que permitía obtener los datos de la simulación inmediatamente, aunque no se mostrase el proceso de la simulación por pantalla.

5

Integración, pruebas y resultados

Para validar la aplicación se han llevado a cabo algunas pruebas creando sistemas que también podían ser creados en una de las aplicaciones estudiadas: QSA. Ambos resultados son contrastados. Con estas pruebas, pese a estar limitadas a un solo servidor, sirven para poder comprobar de manera unitaria las composiciones de sistemas de un solo servidor. Cada uno de los sistemas se ha probado con tres grupos de parámetros preparados para que se obtenga una intensidad de tráfico teórica de 0.9, 0.5 y 0.1. Además, cada simulación en mi sistema se ha ejecutado 5 veces para obtener un resultado medio y una varianza en los resultados. Todos los resultados se pueden ver en el apéndice de resultados.

En el caso de los resultados de QSA hay un problema a la hora de compararlos y se debe a que usan una nomenclatura diferente, pero también añaden una breve explicación, si además comparamos los resultados que obtienen con los cálculos teóricos en un sistema simple podemos ver con qué parámetro debemos compararlo en la aplicación de este trabajo.

Para la mayoría de estas pruebas en las que se compara los resultados con los obtenidos en QSA los componentes que acompañan al servidor son un input y un output. De manera que salvo que se mencione lo contrario estructura de los sistemas creados para las pruebas será de un input que conecta con un servidor que conecta finalmente con el output. Los parámetros del input también serán, salvo que se indique lo contrario, prioridad 1 y sin ciclos de actividad.

5.1. Sistema M | M | 1

Es el sistema más simple que se puede construir, ya que el servidor tiene un único procesador. Los parámetros usados han sido:

Nombre del parámetro	Valor por intensidad de tráfico teórica			Unidades
	0,9	0,5	0,1	
Lambda	1,8	1	1	peticiones por unidad de tiempo
Tasa de entrada al sistema	0,555555555	1	1	unidades de tiempo por petición
Distribución de entrada al sistema	Poisson	Poisson	Poisson	tipo de distribución
Mu	2	2	10	peticiones por unidad de tiempo
Tasa de salida del servidor	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del servidor	Poisson	Poisson	Poisson	tipo de distribución
Máxima cola permitida	Infinita	Infinita	Infinita	peticiones

Tabla 1.1 Parámetros de entrada usados para la prueba de M | M | 1

5.2. Sistema M | M | 2

Con este sistema comprobamos la funcionalidad de un servidor con más de un procesador. En esta prueba se validarán los datos contra el modelo M | M | 2 especificado en QSA, pero se puede generalizar para los modelos M | M | c, siendo c en este caso 2. Los parámetros usados han sido:

Nombre del parámetro	Valor por intensidad de tráfico teórica			Unidades
	0,9	0,5	0,1	
Lambda	1,8	1	1	peticiones por unidad de tiempo
Tasa de entrada al sistema	0,555555555	1	1	unidades de tiempo por petición
Distribución de entrada al sistema	Poisson	Poisson	Poisson	tipo de distribución
Mu	2	2	10	peticiones por unidad de tiempo
Tasa de salida del procesador 1	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del procesador 1	Poisson	Poisson	Poisson	tipo de distribución
Tasa de salida del procesador 2	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del procesador 2	Poisson	Poisson	Poisson	tipo de distribución
Máxima cola permitida	Infinita	Infinita	Infinita	peticiones

Tabla 1.2 Parámetros de entrada usados para la prueba de M | M | 2

5.3. Sistema M | M | 1 | K

Con este sistema comprobamos la funcionalidad de un servidor con una cola limitada. Los parámetros usados han sido:

Nombre del parámetro	Valor por intensidad de tráfico teórica			Unidades
	0,9	0,5	0,1	
Lambda	1,8	1	1	peticiones por unidad de tiempo
Tasa de entrada al sistema	0,555555555	1	1	unidades de tiempo por petición
Distribución de entrada al sistema	Poisson	Poisson	Poisson	tipo de distribución
Mu	2	2	10	peticiones por unidad de tiempo
Tasa de salida del servidor	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del servidor	Poisson	Poisson	Poisson	tipo de distribución
K	5	5	5	peticiones
Máxima cola permitida	4	4	4	peticiones

Tabla 1.3 Parámetros de entrada usados para la prueba de M | M | 1 | K

5.4. Sistema M | M | c | K

Con este sistema comprobamos la funcionalidad de un servidor con varios procesadores y con una cola limitada. Los parámetros usados han sido:

Nombre del parámetro	Valor por intensidad de tráfico teórica			Unidades
	0,9	0,5	0,1	
Lambda	1,8	1	1	peticiones por unidad de tiempo
Tasa de entrada al sistema	0,555555555	1	1	unidades de tiempo por petición
c	2	2	2	procesadores
Tasa de salida del procesador 1	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del procesador 1	Poisson	Poisson	Poisson	tipo de distribución
Tasa de salida del procesador 2	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del procesador 2	Poisson	Poisson	Poisson	tipo de distribución
K	6	6	6	peticiones
Máxima cola permitida	4	4	4	peticiones

Tabla 1.4 Parámetros de entrada usados para la prueba de M | M | c | K

5.5. Sistema M | M | c | c

Con este sistema comprobamos la funcionalidad de un servidor con varios procesadores y sin cola. Los parámetros usados han sido:

Nombre del parámetro	Valor por intensidad de tráfico teórica			Unidades
	0,9	0,5	0,1	
Lambda	1,8	1	1	peticiones por unidad de tiempo
Tasa de entrada al sistema	0,555555555	1	1	unidades de tiempo por petición
Distribución de entrada al sistema	Poisson	Poisson	Poisson	tipo de distribución
c	2	2	2	procesadores
Mu	2	2	10	peticiones por unidad de tiempo
Tasa de salida del procesador 1	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del procesador 1	Poisson	Poisson	Poisson	tipo de distribución
Tasa de salida del procesador 2	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del procesador 2	Poisson	Poisson	Poisson	tipo de distribución
Máxima cola permitida	0	0	0	peticiones

Tabla 1.5 Parámetros de entrada usados para la prueba de M | M | c | c

5.6. Sistema M | Gamma | 1

Una vez comprobada la funcionalidad de sistemas con más de una configuración de procesadores y colas, comprobamos la funcionalidad de otras distribuciones. Los parámetros usados han sido:

Nombre del parámetro	Valor por intensidad de tráfico teórica			Unidades
	0,9	0,5	0,1	
Lambda	1,8	1	1	peticiones por unidad de tiempo
Tasa de entrada al sistema	0,555555555	1	1	unidades de tiempo por petición
Distribución de entrada al sistema	Poisson	Poisson	Poisson	tipo de distribución
Mu	2	2	10	peticiones por unidad de tiempo
Tasa de salida del servidor	0,5	0,5	0,1	unidades de tiempo por petición
Parámetro adicional de Gamma	2	2	2	-
Distribución de salida del servidor	Gamma	Gamma	Gamma	tipo de distribución
Máxima cola permitida	Infinita	Infinita	Infinita	peticiones

Tabla 1.6 Parámetros de entrada usados para la prueba de M | Gamma | 1

5.7. Sistema M | D | 1

También probamos la funcionalidad con una tasa de salida del servidor constante. Los parámetros usados han sido:

Nombre del parámetro	Valor por intensidad de tráfico teórica			Unidades
	0,9	0,5	0,1	
Lambda	1,8	1	1	peticiones por unidad de tiempo
Tasa de entrada al sistema	0,555555555	1	1	unidades de tiempo por petición
Distribución de entrada al sistema	Poisson	Poisson	Poisson	tipo de distribución
Mu	2	2	10	peticiones por unidad de tiempo
Tasa de salida del servidor	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del servidor	Constante	Constante	Constante	tipo de distribución
Máxima cola permitida	Infinita	Infinita	Infinita	peticiones

Tabla 1.7 Parámetros de entrada usados para la prueba de M | D | 1

5.8. Sistema M | M | 1 non Preemptive Priority

Para finalizar las pruebas comparando los resultados con los obtenidos en la aplicación de QSA probamos las prioridades, en este caso hay dos inputs distintos que producen peticiones con distinta prioridad. Los parámetros usados han sido:

Nombre del parámetro	Valor por intensidad de tráfico teórica			Unidades
	0,9	0,5	0,1	
Lambda 1	0,9	0,5	0,5	peticiones por unidad de tiempo
Tasa de entrada al sistema de input 1	1,111111111	2	2	unidades de tiempo por petición
istribución de entrada al sistema de input	Poisson	Poisson	Poisson	tipo de distribución
Prioridad de input 1	1	1	1	prioridad asignada a las peticiones
Lambda 2	0,9	0,5	0,5	peticiones por unidad de tiempo
Tasa de entrada al sistema de input 2	1,111111111	2	2	unidades de tiempo por petición
istribución de entrada al sistema de input	Poisson	Poisson	Poisson	tipo de distribución
Prioridad de input 2	2	2	2	prioridad asignada a las peticiones
Mu	2	2	10	peticiones por unidad de tiempo
Tasa de salida del servidor	0,5	0,5	0,1	unidades de tiempo por petición
Distribución de salida del servidor	Constante	Constante	Constante	tipo de distribución
Máxima cola permitida	Infinita	Infinita	Infinita	peticiones

Tabla 1.8 Parámetros de entrada usados para la prueba de M | M | 1 non Preemptive Priority

Una vez se han validado las simulaciones de sistemas simples se deben validar los sistemas complejos, que son aquellos que tienen retroalimentación, redirección de peticiones rechazadas en algún servidor o balanceadores que distribuyan las peticiones a varios servidores. Para validar estos sistemas se contrastarán los valores resultantes de la simulación con sus valores teóricos calculados con modelos matemáticos simples.

5.9. Sistema complejo con balanceadores

En esta prueba se ha creado un sistema que divide la carga en dos ramas diferentes que se vuelven a unir al final. El objetivo principal de la prueba es comprobar que las tasas de entrada de cada rama son las debidas y que al unir las ramas de nuevo también se obtiene la tasa de entrada debida.

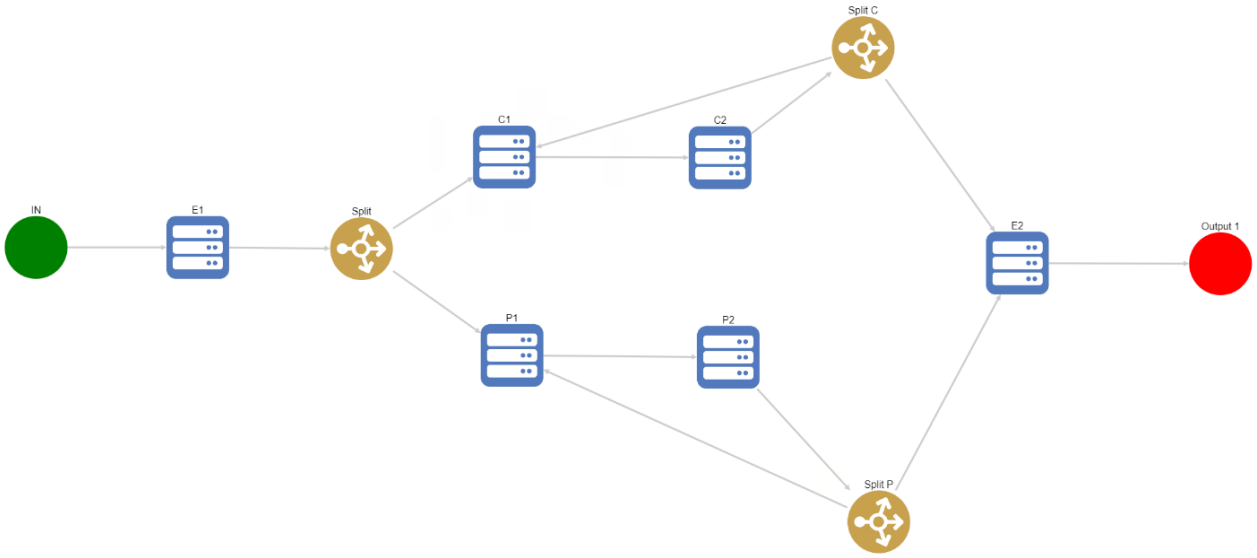

Figura 3.31 Diagrama de sistema complejo con balanceadores

Nombre del parámetro	Valor usado	Unidades
Tasa de entrada del input IN	7	unidades de tiempo por petición
Distribución de entrada del input IN	Poisson	tipo de distribución
Tasa de salida del servidor E1	1	unidades de tiempo por petición
Distribución de salida del servidor E1	Poisson	tipo de distribución
Máxima cola permitida de E1	Infinita	peticiones
Tipo de balanceador de Split	por probabilidad	tipo de balanceador
Probabilidad de dirigir a C1 de Split	0,75	-
Probabilidad de dirigir a P1 de Split	0.25	-
Tasa de salida del servidor C1	4	unidades de tiempo por petición
Distribución de salida del servidor C1	Poisson	tipo de distribución
Máxima cola permitida de C1	Infinita	peticiones
Tasa de salida del servidor C2	2	unidades de tiempo por petición
Distribución de salida del servidor C2	Poisson	tipo de distribución
Máxima cola permitida de C2	Infinita	peticiones
Tipo de balanceador de Split C	por probabilidad	tipo de balanceador
Probabilidad de dirigir a C1 de Split	0,5	-
Probabilidad de dirigir a E2 de Split	0,5	-
Tasa de salida del servidor P1	10	unidades de tiempo por petición
Distribución de salida del servidor P1	Poisson	tipo de distribución
Máxima cola permitida de P1	Infinita	peticiones
Tasa de salida del servidor P2	4	unidades de tiempo por petición
Distribución de salida del servidor P2	Poisson	tipo de distribución
Máxima cola permitida de P2	Infinita	peticiones
Tipo de balanceador de Split P	por probabilidad	tipo de balanceador
Probabilidad de dirigir a P1 de Split	0,25	-
Probabilidad de dirigir a E2 de Split	0,75	-
Tasa de salida del servidor E2	0,25	unidades de tiempo por petición
Distribución de salida del servidor E2	Poisson	tipo de distribución
Máxima cola permitida de E2	Infinita	peticiones

Tabla 1.9 Parámetros de entrada usados para el sistema complejo con balanceadores

5.10. Sistema complejo con retroalimentación

En esta prueba se ha creado un sistema con una rama que vuelve al inicio del sistema, creando así un sistema con retroalimentación. El objetivo principal de la prueba es comprobar que el primer servidor, que recibe la retroalimentación, actúa como debe comparándolo con los valores teóricos.

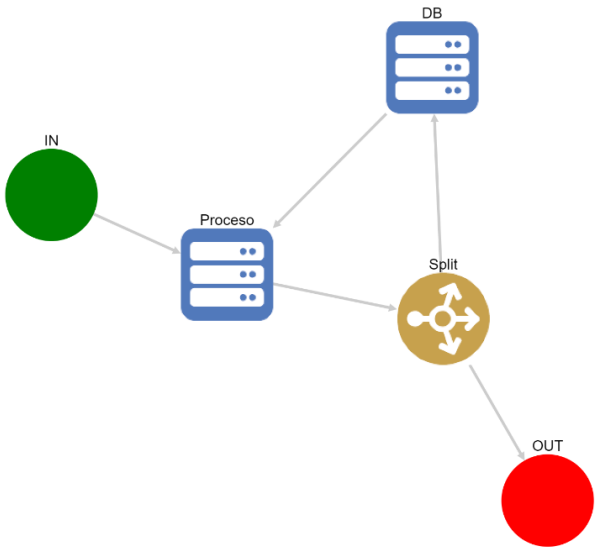

Figura 3.32 Diagrama de sistema complejo con retroalimentación

Nombre del parámetro	Valor usado	Unidades
Distribución de entrada del input IN	Poisson	tipo de distribución
Tasa de salida del servidor Proceso	2	unidades de tiempo por petición
Distribución de salida del servidor Proceso	Poisson	tipo de distribución
Máxima cola permitida de Proceso	Infinita	peticiones
Tasa de salida del servidor DB	2,5	unidades de tiempo por petición
Distribución de salida del servidor DB	Poisson	tipo de distribución
Máxima cola permitida de DB	Infinita	peticiones
Tipo de balanceador de Split	por probabilidad	tipo de balanceador
Probabilidad de dirigir a DB de Split	0,5	-
Probabilidad de dirigir a OUT de Split	0,5	-

Tabla 1.10 Parámetros de entrada usados para el sistema complejo con retroalimentación

5.11. Sistema complejo con redirección de peticiones rechazadas

En esta prueba se ha creado un sistema con una rama cuya entrada es exclusivamente las peticiones rechazadas por el primer servidor. El objetivo principal de esta prueba es comprobar el comportamiento del servidor que recibe las peticiones rechazadas, sobre todo su tasa de entrada, comparándola con los valores teóricos.

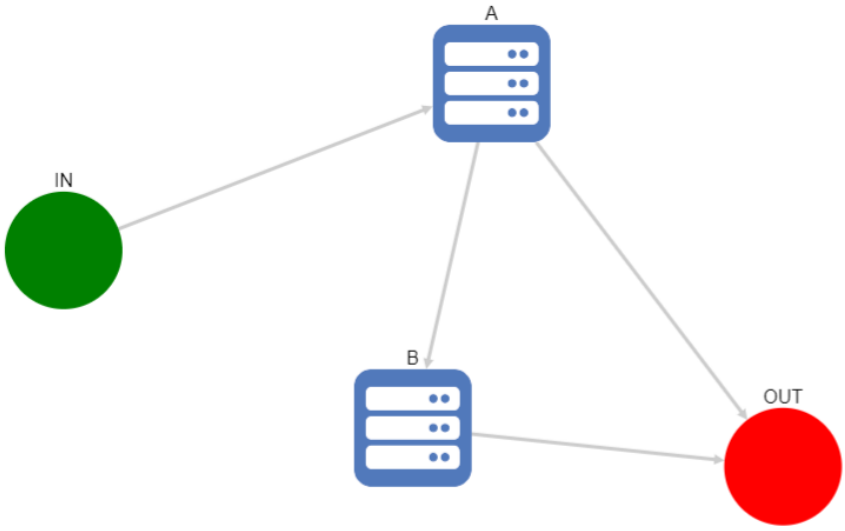

Figura 3.33 Diagrama de sistema complejo con redirección de peticiones rechazadas

Nombre del parámetro	Valor usado	Unidades
Tasa de entrada del input IN	0,111111111	unidades de tiempo por petición
Distribución de entrada del input IN	Poisson	tipo de distribución
Tasa de salida del servidor A	0,1	unidades de tiempo por petición
Distribución de salida del servidor A	Poisson	tipo de distribución
Máxima cola permitida de A	5	peticiones
Target de peticiones rechazadas de A	B	Componente
Tasa de salida del servidor B	0,5	unidades de tiempo por petición
Distribución de salida del servidor B	Poisson	tipo de distribución
Máxima cola permitida de B	Infinita	peticiones

Tabla 1.11 Parámetros de entrada usados para el sistema complejo con redirección de peticiones rechazadas

5.12. Sistema complejo con picos de trabajo

En esta prueba se ha creado un sistema con dos inputs, uno de ellos será el que introduzca los picos de trabajo al sistema teniendo periodos de actividad e inactividad. Esta característica no es analizable por los modelos matemáticos simples empleados en las anteriores pruebas. Sin embargo, si se podrá observar el correcto funcionamiento de los periodos de actividad e inactividad del segundo input. Para validar esta característica se ha creado un sistema con un único servidor cuya tasa de salida es suficiente para procesar las peticiones sin acumularlas durante el periodo inactivo del segundo input, pero que comience a acumular peticiones durante el periodo activo del segundo input. De esta manera, durante la ejecución se deben ver los periodos y en los valores finales de la simulación deben darse el caso de que el número medio de peticiones en el servidor sea un número intermedio entre 0 y el valor máximo alcanzado por la cola. De la misma manera la tasa de entrada al servidor calculada debe ser un valor intermedio entre la tasa conjunta de los dos inputs y la tasa del input principal.

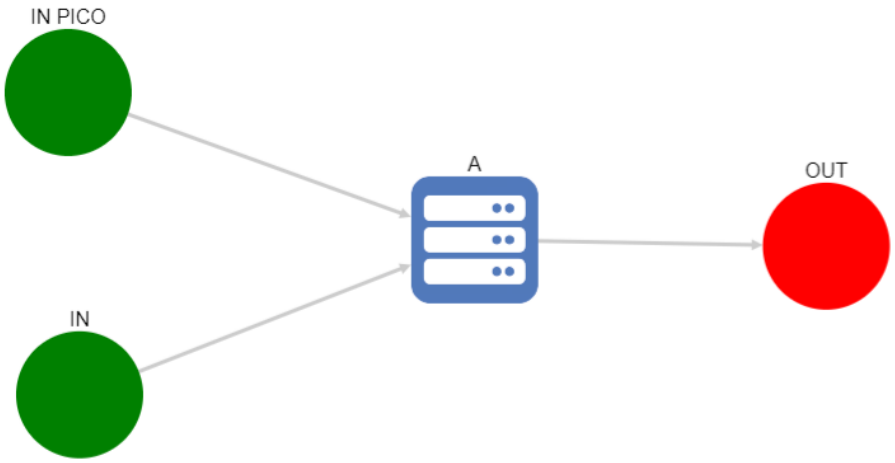

Figura 3.34 Diagrama de sistema complejo con picos de trabajo

Nombre del parámetro	Valor usado	Unidades
Tasa de entrada del input IN	0,4	unidades de tiempo por petición
Distribución de entrada del input IN	Poisson	tipo de distribución
Tasa de entrada del input IN PICO	0,1	unidades de tiempo por petición
Distribución de entrada del input IN PICO	Poisson	tipo de distribución
Comienza activo	Sí	-
Duración del ciclo activo del input IN PICO	50	unidades de tiempo
Duración del ciclo inactivo del input IN PICO	100	unidades de tiempo
Tasa de salida del servidor A	0,15	unidades de tiempo por petición
Distribución de salida del servidor A	Poisson	tipo de distribución
Máxima cola permitida de A	Infinita	peticiones

Tabla 1.12 Parámetros de entrada usados para el sistema complejo con picos de trabajo

5.13. Sistema complejo no analizable por modelos matemáticos simples

La última prueba para validar la aplicación es una en la que se compruebe la capacidad de simular sistemas no analizables por modelos matemáticos simples, para ello se creó el sistema mostrado. En este sistema se combinan varias partes de los sistemas validados previamente. Además de introducir los picos de trabajo que se validarán en esta prueba, debido a que no se pueden analizar picos de trabajo con los modelos matemáticos simples. Las partes del sistema son:

- **Picos de trabajo**: El sistema tiene un input principal que se mantiene activo durante toda la simulación, y, además, tiene un input secundario que simulará los picos de trabajo teniendo un periodo durante el que estará activo (en este sistema este periodo dura 50 unidades de tiempo) y otro durante el que estará inactivo (en este sistema este periodo dura 100 unidades de tiempo). El input secundario alternará entre estos dos periodos durante toda la simulación, haciendo que la entrada de peticiones al sistema sea la suma de ambos inputs durante el periodo activo y durante el periodo inactivo sólo se generen peticiones desde el input principal.

Las tasas de entrada al sistema de estos inputs esta seleccionada de forma que durante el periodo activo los servidores A y B tendrán una tasa de salida insuficiente, provocando que sus colas se llenen, y durante el periodo inactivo podrán procesar las peticiones almacenadas en la cola.

- **Retroalimentación**: El sistema tiene dos retroalimentaciones. Una de ellas reintroduce la mitad de las peticiones que entran al servidor LOG en el servidor BAL. La otra retroalimentación es más corta, ya que reintroduce un cuarto de las peticiones procesadas por el servidor C en el servidor C.

- **Redirección de peticiones rechazadas**: El servidor C tiene como entrada de peticiones, a parte de la ya comentada retroalimentación, las peticiones rechazadas por el servidor B. Por tanto, el servidor B tiene una cola limitada para que al llenarse durante el periodo activo del input secundario comience a rechazar peticiones.

- **Ramificaciones creadas por balanceadores**: A parte de los balanceadores explicados en la retroalimentación hay dos balanceadores más: El balanceador BAL S distribuye las peticiones procesadas por el servidor BAL entre los servidores A y B con la misma probabilidad, es decir, la mitad de las peticiones se enviarán al servidor A y la otra mitad al servidor B. El otro balanceador es Split P que enviará un cuarto de las peticiones procesadas por los servidores A, B y C al servidor LOG y el resto al output OUT.

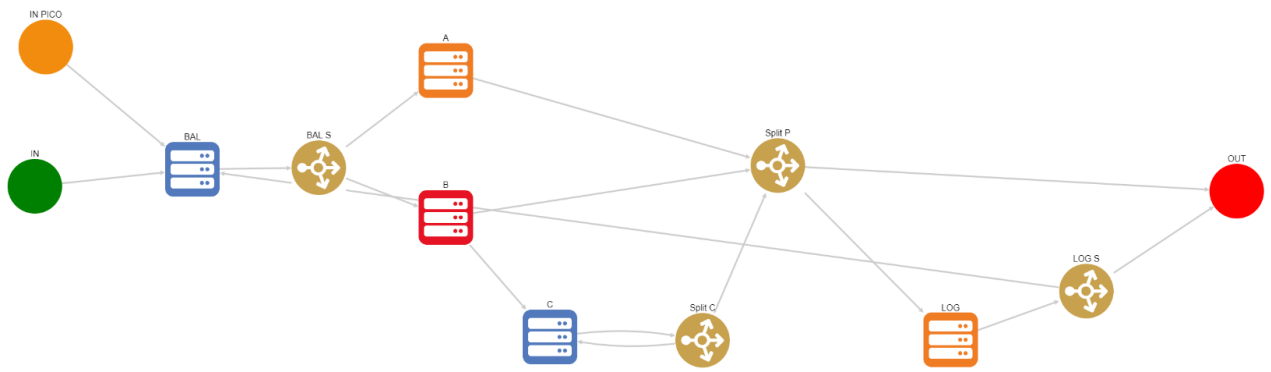

Figura 3.35 Diagrama de sistema complejo no analizable por modelos matemáticos simples

Dado que este sistema no es analizable por modelos matemáticos simples no hay valores teóricos con los que contrastar los valores obtenidos en la simulación. Sin embargo, en esta prueba no se está tratando de validar un sistema, ya que se han validado previamente las partes individuales del mismo. El objetivo de esta prueba es demostrar que la aplicación es capaz de simular un sistema que no es analizable mediante modelos matemáticos simples.

Nombre del parámetro	Valor usado	Unidades
Tasa de entrada del input IN PICO	0,1	unidades de tiempo por petición
Comienza activo	Sí	-
Distribución de entrada del input IN PICO	Poisson	tipo de distribución
Duración del ciclo activo del input IN PICO	50	unidades de tiempo
Duración del ciclo inactivo del input IN PICO	100	unidades de tiempo
Tasa de entrada del input IN	0,3	unidades de tiempo por petición
Distribución de entrada del input IN	Poisson	tipo de distribución
Tasa de salida del servidor BAL	0,05	unidades de tiempo por petición
Distribución de salida del servidor BAL	Poisson	tipo de distribución
Máxima cola permitida de BAL	Infinita	peticiones
Tipo de balanceador de BAL S	por probabilidad	tipo de balanceador
Probabilidad de dirigir a A de BAL S	0,5	-
Probabilidad de dirigir a B de BAL S	0,5	-
Tasa de salida del servidor A	0,2	unidades de tiempo por petición
Distribución de salida del servidor A	Poisson	tipo de distribución
Máxima cola permitida de A	Infinita	peticiones
Tasa de salida del servidor B	0,2	unidades de tiempo por petición
Distribución de salida del servidor B	Poisson	tipo de distribución
Máxima cola permitida de B	5	peticiones
Target de peticiones rechazadas de B	C	Componente
Tasa de salida del servidor C	0,1	unidades de tiempo por petición
Distribución de salida del servidor C	Poisson	tipo de distribución
Máxima cola permitida de C	Infinita	peticiones
Tipo de balanceador de Split C	por probabilidad	tipo de balanceador
Probabilidad de dirigir a C de Split C	0,25	-
Probabilidad de dirigir a Split P de Split C	0,75	-
Tipo de balanceador de Split P	por probabilidad	tipo de balanceador
Probabilidad de dirigir a LOG de Split P	0,75	-
Probabilidad de dirigir a OUT de Split P	0,25	-
Tasa de salida del servidor LOG	0,125	unidades de tiempo por petición
Distribución de salida del servidor LOG	Constante	tipo de distribución
Máxima cola permitida de LOG	Infinita	peticiones
Tipo de balanceador de LOG S	por probabilidad	tipo de balanceador
Probabilidad de dirigir a BAL de LOG S	0,5	-
Probabilidad de dirigir a OUT de LOG S	0,5	-

Tabla 1.13 Parámetros de entrada usados para el sistema complejo no analizable por modelos matemáticos simples

6

Conclusiones y trabajo futuro

6.1. Conclusiones

Crear una interfaz gráfica es fundamentalmente diferente a la programación que hemos estudiado durante la carrera, sobre todo cuando usas HTML y Css, dos lenguajes no muy usados en la carrera. También cabe destacar que React tiene elementos diferentes que hay que aprender, como por ejemplo los ciclos de vida en React. Sin embargo, como con cualquier otro lenguaje o framework nuevo se puede aprender y mejorar en su uso al empaparte de él y conocer la forma en la que está pensado su uso. Además, el comprender de manera más profunda los problemas y soluciones implícitos en el desarrollo de aplicaciones Web y sobre todo de la interfaz gráfica en la parte del cliente es algo muy beneficioso para mí.

La experiencia de implementar una aplicación con gran énfasis en la parte del cliente es muy interesante. Cuanto más te alejas del backend más te das cuenta de que el objetivo del programa es distinto. En la mayoría de las asignaturas de la carrera hemos implementado programas que no debían esperar mucha interactividad con el usuario y por lo tanto requisitos como la usabilidad o la accesibilidad no eran muy prioritarios o restrictivos. Sin embargo, en este caso la usabilidad si es un requisito a tener en cuenta. Además, en el caso de esta aplicación al estar implementada en React se debe uno adecuar a las formas en las que actúa con sus ciclos de vida y las condiciones bajo las cuales se actualiza la página.

Otra experiencia interesante es la de programar una aplicación entera en JavaScript, ya que pese a sus similitudes con Java plantea un entorno distinto donde las variables no están tipadas haciendo el código más ligero no obligándote a especificar el tipo de cada variable. Pero, la falta de tipo en las variables también puede llevar a problemas inesperados cuando esperas un tipo de valor y recibes otro, complicado el proceso de detección de errores.

En resumen, el crear esta aplicación no ha resultado sencillo debido a todas las tecnologías y problemas encontrados que son esencialmente diferentes a los encontrados durante la carrera, pero, el hecho de haber aprendido a manejarlos es beneficioso para mí. Durante toda la carrera se nos han enseñado varias ramas de la informática para que como Ingenieros Informáticos podamos manejarnos en cualquier entorno, ya que no es extraño que, aunque en el futuro no sea necesario que comprenda al completo estos entornos, deba tratar con otras personas, aplicaciones o problemas que requieran cierto nivel de conocimientos en muchos campos distintos. Sin ir más lejos hay muchas empresas de informática que desarrollan aplicaciones Web, haciendo de esta experiencia una buena apuesta a futuro.

6.2. Trabajo futuro

Como todo sistema, aplicación o programa, este proyecto tiene sus limitaciones y posibilidades de mejora.

Una limitación clara es la necesidad de mantener la aplicación en primer plano para mantener la ejecución. Esto se debe a que la simulación se ejecuta localmente en el navegador, por tanto, una solución sería mover esta ejecución al lado del servidor. Esto sería sencillo de hacer, ya que el modelo que contiene el código que simula está separado de la vista. Sin embargo, esta solución solo serviría para la ejecución rápida sin mostrar gráficamente la evolución, para este tipo de ejecución solo hay una petición y una única respuesta. El problema sería con la ejecución gráfica que muestra la evolución de la simulación gráficamente, en este caso habría que mantener una conexión activa con el cliente durante toda la simulación. Esto es un problema ya que uno de los objetivos es usar la aplicación como apoyo a la hora de aprender los modelos matemáticos simples y por tanto la escalabilidad de la aplicación es algo a tener en cuenta.

Otra limitación serían los tipos de distribuciones disponibles en la aplicación. Estos, por la forma en la que está pensado el código, serían fáciles de añadir, solo hace falta crear una función que genere un número de la distribución dada por cada distribución que se quiera añadir. Sin ir más lejos en la librería Random hay muchas distribuciones que están marcadas como futuras adicciones que podrían ser añadidas también a la aplicación.

Como última limitación hay que destacar la detección de errores, que pese a detectar errores simples que impedirían claramente la ejecución de la simulación, no es capaz de detectar bucles infinitos u otros sistemas que presenten errores similares.

Referencias

[1] Wainer, G., & Mosterman, P. (2011). Discrete-event modeling and simulation : Theory and applications (Computational analysis, synthesis, and design of dynamic systems). Boca Raton; London; New York: CRC Press, Taylor & Francis Group.

[2] Sztrik, J. (2016). Basic Queueing Theory. (añadida Mayo 17, 2021) http://irh.inf.unideb.hu/user/jsztrik

[3] Stellman, Andrew, & Greene, Jennifer. (2005). Applied Software Project Management. Sebastopol: O'Reilly Media, Incorporated.

[4] M. Glinz. (2007). "On Non-Functional Requirements," 15th IEEE International Requirements Engineering Conference (RE 2007), 2007, pp. 21-26, doi: 10.1109/RE.2007.45.

[5] Wiegers, K. (2009). Software Requirements (2nd ed.). Sebastopol: Microsoft Press.

[6] Pazos Arias, J., Suárez González, A., & Díaz Redondo, R. (2003). Teoría de colas y simulación de eventos discretos. Madrid [etc.: Pearson Educación.

[7] Guía docente Sistemas Informáticos II (añadida Mayo 16, 2021) http://www.uam.es/EPS/GuiasDocentesGrado/1242700889012.htm?idenlace=1446764777645&language=es&nodepath=Gu?as%20Docentes

[8] QSA Applications. University of Debrecen (Hungary) (añadida Mayo 16,2021)

https://qsa.inf.unideb.hu/

[9] Kendall's Notation. (2013). Encyclopedia of Operations Research and Management, Encyclopedia of Operations Research and Management, 2013.

[10] Java Modelling Tools (añadida Mayo 16, 2021)

http://jmt.sourceforge.net/

[11] Heroku (añadida Mayo 15, 2021)

[12] Aplicación creada en este trabajo de fin de grado (añadida Mayo 15, 2021) https://tfg-production-app.herokuapp.com/

[13] David Griffiths, & Dawn Griffiths. (2021). React Cookbook. O'Reilly Media.

[14] Chinnathambi, K. (2018). Learning React : : A hands-on guide to building web applications using React and Redux / (Second ed.).

[15] Angular (añadida Mayo 15, 2021) https://angular.io/

[16] Node Js (añadida Mayo 16, 2021) https://nodejs.org/es/

[17] NPM (añadida Mayo 16, 2021) https://www.npmjs.com/

[18] GoJs (añadida Mayo 16, 2021) https://gojs.net/latest/index.html

[19] Cytoscape.js (añadida Mayo 16, 2021) https://js.cytoscape.org/

[20] Max Franz, Christian T. Lopes, Gerardo Huck, Yue Dong, Onur Sumer, Gary D. Bader. (2016) Cytoscape.js: a graph theory library for visualisation and analysis, Bioinformatics, Volume 32, Issue 2, 15 January 2016, Pages 309–311, https://doi.org/10.1093/bioinformatics/btv557

[21] Random (añadida Mayo 16, 2021) https://www.npmjs.com/package/random

[22] D3 (añadida Mayo 16, 2021) https://github.com/d3/d3-random

[23] Fontawesome (añadida Mayo 16, 2021) https://fontawesome.com/

[24] Reactstrap (añadida Mayo 16, 2021) https://reactstrap.github.io/

[25] Bootstrap (añadida Mayo 16, 2021) https://getbootstrap.com/

[26] Yanette Díaz González, & Yenisleidy Fernández Romero. (2012). Patrón Modelo-Vista-Controlador. Telem@tica (La Habana), 11(1), 47-57.

[27] Create-react-app (añadida Mayo 17, 2021) https://es.reactjs.org/docs/create-a-new-react-app.html

[28] Plugin de Google Chrome para React (añadida Mayo 17, 2021) https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=es

[29] Green, I., Loukides, M., St. Laurent, S., & Romano, R. (2012). Web workers / (First ed.).

Apéndices

I A

Resultados de las pruebas

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,9	0,9009	0,0106	-
N	L	9	8,7403	1,5034	peticiones
Q	Lq	8,1	7,8395	1,4935	peticiones
T	W	5	4,8505	0,8100	unidades de tiempo
W	Wq	4,5	4,3477	0,8080	unidades de tiempo

Tabla 1.14 Resultados de la prueba de M | M | 1 para intensidad de tráfico teórica 0,9

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,5	0,5055	0,0038	-
N	L	1	1,0257	0,0168	peticiones
Q	Lq	0,5	0,5203	0,0152	peticiones
T	W	1	1,0173	0,0097	unidades de tiempo
W	Wq	0,5	0,5159	0,0112	unidades de tiempo

Tabla 1.15 Resultados de la prueba de M | M | 1 para intensidad de tráfico teórica 0,5

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,1	0,0997	0,0020	-
N	L	0,111	0,1110	0,0029	peticiones
Q	Lq	0,0111	0,0113	0,0009	peticiones
T	W	0,111	0,1111	0,0011	unidades de tiempo
W	Wq	0,0111	0,0113	0,0008	unidades de tiempo

Tabla 1.16 Resultados de la prueba de M | M | 1 para intensidad de tráfico teórica 0,1

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,621	0,6191	0,0035	-
N	L	1,13	1,1266	0,0292	peticiones
Q	Lq	0,229	0,2298	0,0196	peticiones
T	W	0,627	0,6263	0,0132	unidades de tiempo
W	Wq	0,127	0,1277	0,0102	unidades de tiempo

Tabla 1.17 Resultados de la prueba de M | M | 2 para intensidad de tráfico teórica 0,9

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,4	0,3980	0,0044	-
N	L	0,533	0,5284	0,0060	peticiones
Q	Lq	0,0333	0,0322	0,0023	peticiones
T	W	0,533	0,5307	0,0060	unidades de tiempo
W	Wq	0,0333	0,0324	0,0022	unidades de tiempo

Tabla 1.18 Resultados de la prueba de M | M | 2 para intensidad de tráfico teórica 0,5

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,0952	0,0942	0,0011	-
N	L	0.1	0,0995	0,0012	peticiones
Q	Lq	0	0,0003	0,0000	peticiones
T	W	0,1	0,0994	0,0008	unidades de tiempo
W	Wq	0	0,0003	0,0000	unidades de tiempo

Tabla 1.19 Resultados de la prueba de M | M | 2 para intensidad de tráfico teórica 0,1

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,787	0,7841	0,0057	-
N	L	2,19	2,1837	0,0328	peticiones
Q	Lq	1,41	1,3997	0,0275	peticiones
T	W	1,4	1,3911	0,0265	unidades de tiempo
W	Wq	0,895	0,7803	0,0140	unidades de tiempo

Tabla 1.20 Resultados de la prueba de M | M | 1 | K para intensidad de tráfico teórica 0,9

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,492	0,4932	0,0078	-
N	L	0,905	0,9060	0,0213	peticiones
Q	Lq	0,413	0,4128	0,0137	peticiones
T	W	0,919	0,9207	0,0187	unidades de tiempo
W	Wq	0,419	0,4133	0,0125	unidades de tiempo

Tabla 1.21 Resultados de la prueba de M | M | 1 | K para intensidad de tráfico teórica 0,5

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,1	0,0993	0,0013	-
N	L	0,111	0,1103	0,0017	peticiones
Q	Lq	0,0111	0,0109	0,0005	peticiones
T	W	0,111	0,1107	0,0013	unidades de tiempo
W	Wq	0,0111	0,0110	0,0005	unidades de tiempo

Tabla 1.22 Resultados de la prueba de M | M | 1 | K para intensidad de tráfico teórica 0,1

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,619	0,6201	0,0012	-
N	L	1,09	1,0994	0,0073	peticiones
Q	Lq	0,2	0,2016	0,0069	peticiones
T	W	0,612	0,6137	0,0037	unidades de tiempo
W	Wq	0,112	0,1125	0,0038	unidades de tiempo

Tabla 1.23 Resultados de la prueba de M | M | c | K para intensidad de tráfico teórica 0,9

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,4	0,4020	0,0034	-
N	L	0,533	0,5393	0,0083	peticiones
Q	Lq	0,0328	0,0349	0,0030	peticiones
T	W	0,533	0,5380	0,0069	unidades de tiempo
W	Wq	0,0328	0,0348	0,0029	unidades de tiempo

Tabla 1.24 Resultados de la prueba de M | M | c | K para intensidad de tráfico teórica 0,5

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,0952	0,0962	0,0011	-
N	L	0,1	0,1014	0,0014	peticiones
Q	Lq	0	0,0002	0,0000	peticiones
T	W	0,1	0,1011	0,0014	unidades de tiempo
W	Wq	0	0,0002	0,0000	unidades de tiempo

Tabla 1.25 Resultados de la prueba de M | M | c | K para intensidad de tráfico teórica 0,1

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,566	0,5661	0,0043	-
N	L	0,742	0,7414	0,0073	peticiones
W	Wq	0,5	0,5004	0,0037	unidades de tiempo

Tabla 1.26 Resultados de la prueba de M | M | c | c para intensidad de tráfico teórica 0,9

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,385	0,3851	0,0034	-
N	L	0,462	0,4614	0,0038	peticiones
W	Wq	0,5	0,5007	0,0060	unidades de tiempo

Tabla 1.27 Resultados de la prueba de M | M | c | c para intensidad de tráfico teórica 0,5

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
Us	Rho	0,095	0,0943	0,0006	-
N	L	0,0995	0,0990	0,0006	peticiones
T	W	0,1	0,0993	0,0003	unidades de tiempo

Tabla 1.28 Resultados de la prueba de M | M | c | c para intensidad de tráfico teórica 0,1

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
ρ	Rho	0,9	0,9026	0,0147	-
N	L	6,98	6,8482	0,9762	peticiones
Q	Lq	6,08	5,9456	0,9630	peticiones
T	W	3,88	3,7971	0,5080	unidades de tiempo
W	Wq	3,38	3,2951	0,5056	unidades de tiempo

Tabla 1.29 Resultados de la prueba de M | Gamma | 1 para intensidad de tráfico teórica 0,9

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
ρ	Rho	0,5	0,5038	0,0143	-
N	L	0,875	0,8884	0,0291	peticiones
Q	Lq	0,375	0,3849	0,0156	peticiones
T	W	1,75	1,7652	0,0280	unidades de tiempo
W	Wq	0,75	0,7646	0,0191	unidades de tiempo

Tabla 1.30 Resultados de la prueba de M | Gamma | 1 para intensidad de tráfico teórica 0,5

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
ρ	Rho	0,1	0,0998	0,0013	-
N	L	0,108	0,1082	0,0013	peticiones
Q	Lq	0,0083	0,0084	0,0002	peticiones
T	W	0,108	0,1081	0,0004	unidades de tiempo
W	Wq	0,00833	0,0083	0,0002	unidades de tiempo

Tabla 1.31 Resultados de la prueba de M | Gamma | 1 para intensidad de tráfico teórica 0,1

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
ρ	Rho	0,9	0,8963	0,0069	-
N	L	4,95	4,8525	0,3981	peticiones
Q	Lq	4,05	3,9563	0,3939	peticiones
T	W	5,5	5,4104	0,4204	unidades de tiempo
W	Wq	4,5	4,4076	0,4198	unidades de tiempo

Tabla 1.32 Resultados de la prueba de M | D | 1 para intensidad de tráfico teórica 0,9

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
ρ	Rho	0,5	0,4955	0,0065	-
N	L	0,75	0,7313	0,0085	peticiones
Q	Lq	0,25	0,2360	0,0056	peticiones
T	W	1,5	1,4767	0,0126	unidades de tiempo
W	Wq	0,5	0,4767	0,0126	unidades de tiempo

Tabla 1.33 Resultados de la prueba de M | D | 1 para intensidad de tráfico teórica 0,5

Nombre del parámetro en QSA	Nombre del parámetro obtenido	Valor obtenido en QSA	Valor medio obtenido	Varianza obtenida	Unidades
ρ	Rho	0,1	0,1003	0,0012	-
N	L	0,106	0,1058	0,0013	peticiones
Q	Lq	0,0055	0,0056	0,0002	peticiones
T	W	0,106	0,1055	0,0002	unidades de tiempo
W	Wq	0,0055	0,0055	0,0002	unidades de tiempo

Tabla 1.34 Resultados de la prueba de M | D | 1 para intensidad de tráfico teórica 0,1

Nombre del parámetro	Valor teórico	Valor medio obtenido	Varianza obtenida	Unidades
L total	8,1009	8,6641	2,3182	peticiones
Tasa de entrada de IN	7	7,0179	0,3384	unidades de tiempo por petición
Tasa de llegada a E1	7	7,0116	0,3389	unidades de tiempo por petición
Tasa de salida en E1	1	1,0041	0,0334	unidades de tiempo por petición
Lambda de E1	0,1428	0,1429	0,0067	peticiones por unidad de tiempo
Mu de E1	1	0,9968	0,0335	peticiones por unidad de tiempo
Rho de E1	0,1428	0,1436	0,0108	-
L de E1	0,1666	0,1682	0,0140	peticiones
Lq de E1	0,0238	0,0247	0,0043	peticiones
W de E1	1,1667	1,1766	0,0446	unidades de tiempo
Wq de E1	0,1427	0,1725	0,0247	unidades de tiempo
Tasa de llegada a Split	7	7,0128	0,3384	unidades de tiempo por petición
Tasa de llegada a C1	4,6667	4,6838	0,1923	unidades de tiempo por petición
Tasa de salida en C1	4	4,0649	0,1046	unidades de tiempo por petición
Lambda de C1	0,2142	0,2138	0,0087	peticiones por unidad de tiempo
Mu de C1	0,25	0,2461	0,0063	peticiones por unidad de tiempo
Rho de C1	0,8568	0,8650	0,0392	-
L de C1	6,0028	6,5669	2,2363	peticiones
Lq de C1	5,146	5,7023	2,2014	peticiones
W de C1	27,9329	30,5498	9,5594	unidades de tiempo
Wq de C1	23,9329	26,3851	9,5212	unidades de tiempo
Tasa de llegada a C2	4,6667	4,7047	0,1889	unidades de tiempo por petición
Tasa de salida en C2	2	2,0244	0,0628	unidades de tiempo por petición
Lambda de C2	0,2142	0,2128	0,0085	peticiones por unidad de tiempo
Mu de C2	0,5	0,4943	0,0147	peticiones por unidad de tiempo
Rho de C2	0,4284	0,4308	0,0222	-
L de C2	0,7501	0,7500	0,0627	peticiones
Lq de C2	0,3217	0,3194	0,0442	peticiones
W de C2	3,5018	3,5217	0,1619	unidades de tiempo
Wq de C2	1,5018	1,4972	0,1493	unidades de tiempo
Tasa de llegada a P1	21	20,9567	1,9107	unidades de tiempo por petición
Tasa de salida en P1	10	9,6774	0,5205	unidades de tiempo por petición
Lambda de P1	0,0476	0,0480	0,0044	peticiones por unidad de tiempo
Mu de P1	0,1	0,1036	0,0057	peticiones por unidad de tiempo
Rho de P1	0,476	0,4654	0,0619	-
L de P1	0,9091	0,8930	0,2114	peticiones
Lq de P1	0,4331	0,4297	0,1527	peticiones
W de P1	19,0987	18,5274	2,7658	unidades de tiempo
Wq de P1	9,0987	8,8501	2,3590	unidades de tiempo
Tasa de llegada a P2	21	20,9879	1,9201	unidades de tiempo por petición
Tasa de salida en P2	4	4,1153	0,2544	unidades de tiempo por petición
Lambda de P2	0,0476	0,0480	0,0044	peticiones por unidad de tiempo
Mu de P2	0,25	0,2437	0,0149	peticiones por unidad de tiempo
Rho de P2	0,1904	0,1974	0,0247	-
L de P2	0,2352	0,2488	0,0377	peticiones
Lq de P2	0,0448	0,0524	0,0182	peticiones
W de P2	4,9411	5,2022	0,4485	unidades de tiempo
Wq de P2	0,9411	1,0856	0,3184	unidades de tiempo
Tasa de llegada a Split C	4,6667	4,7055	0,1892	unidades de tiempo por petición
Tasa de llegada a Split P	21	21,0061	1,9249	unidades de tiempo por petición
Tasa de llegada a E2	7	7,0609	0,3291	unidades de tiempo por petición
Tasa de salida en E2	0,25	0,2524	0,0087	unidades de tiempo por petición
Lambda de E2	0,1428	0,1419	0,0064	peticiones por unidad de tiempo
Mu de E2	4	3,9661	0,1362	peticiones por unidad de tiempo
Rho de E2	0,0357	0,0358	0,0017	-
L de E2	0,037	0,0373	0,0019	peticiones
Lq de E2	0,0013	0,0015	0,0003	peticiones
W de E2	0,2593	0,2631	0,0098	unidades de tiempo
Wq de E2	0,0093	0,0107	0,0018	unidades de tiempo
W medio total de OUT	56,7064	60,1806	14,6451	unidades de tiempo

Tabla 1.35 Resultados de la prueba de balanceadores

Nombre del parámetro	Valor teórico	Valor medio obtenido	Varianza obtenida	Unidades
L total	5	4,9622	1,0794	peticiones
Tasa de entrada de IN	5	5,0106	0,1839	unidades de tiempo por petición
Tasa de llegada a Proceso	2,5	2,5091	0,1188	unidades de tiempo por petición
Tasa de salida en Proceso	2	2,0067	0,0350	unidades de tiempo por petición
Lambda de Proceso	0,4	0,3992	0,0179	peticiones por unidad de tiempo
Mu de Proceso	0,5	0,4985	0,0087	peticiones por unidad de tiempo
Rho de Proceso	0,8	0,7994	0,0432	-
L de Proceso	4	3,9964	0,9830	peticiones
Lq de Proceso	3,2	3,1971	0,9409	peticiones
W de Proceso	10	9,9342	2,1316	unidades de tiempo
Wq de Proceso	8	7,9115	2,0941	unidades de tiempo
Tasa de llegada a Split	2,5	2,5151	0,1164	
Tasa de llegada a DB	5	5,0215	0,3074	unidades de tiempo por petición
Tasa de salida en DB	2,5	2,4678	0,0687	unidades de tiempo por petición
Lambda de DB	0,2	0,1997	0,0117	peticiones por unidad de tiempo
Mu de DB	0,4	0,4055	0,0110	peticiones por unidad de tiempo
Rho de DB	0,5	0,4922	0,0339	-
L de DB	1	0,9658	0,1284	peticiones
Lq de DB	0,5	0,4736	0,0957	peticiones
W de DB	5	4,8163	0,3934	unidades de tiempo
Wq de DB	2,5	2,3476	0,3551	unidades de tiempo
W medio total de OUT	25	24,6037	4,8259	unidades de tiempo

Tabla 1.36 Resultados de la prueba de retroalimentación

Nombre del parámetro	Valor teórico	Valor medio obtenido	Varianza obtenida	Unidades
L total	3,436	4,8801	0,2796	peticiones
Tasa de entrada de IN	0,1111	0,1108	0,0003	unidades de tiempo por petición
Tasa de salida en A	0,1	0,1002	0,0005	unidades de tiempo por petición
Mu de A	10	9,9786	0,0506	unidades de tiempo por petición
Rho de A	0,8082	0,8113	0,0022	peticiones por unidad de tiempo
L de A	2,588	2,6004	0,0226	peticiones por unidad de tiempo
Lq de A	3,2021	1,7892	0,0209	-
W de A	0,32	0,3212	0,0038	peticiones
Wq de A	0,22	0,1983	0,0024	peticiones
Tasa de llegada a B	1,0893	1,0789	0,0304	unidades de tiempo por petición
Tasa de salida en B	0,5	0,5021	0,0049	unidades de tiempo por petición
Lambda de B	0,918	0,9275	0,0261	peticiones por unidad de tiempo
Mu de B	2	1,9918	0,0194	peticiones por unidad de tiempo
Rho de B	0,459	0,4656	0,0168	-
L de B	0,848	2,2797	0,2617	peticiones
Lq de B	0,389	1,8142	0,2456	peticiones
W de B	0,924	2,4547	0,2187	unidades de tiempo
Wq de B	0,424	1,9522	0,2148	unidades de tiempo
W medio total de OUT	25	24,6037	4,8259	unidades de tiempo

Tabla 1.37 Resultados de la prueba de redirección de peticiones rechazadas

Nombre del parámetro	Valor medio obtenido	Varianza obtenida	Unidades
Tasa de entrada de IN	0,3997	0,0026	unidades de tiempo por petición
Tasa de entrada de IN PICO	0,1001	0,0004	unidades de tiempo por petición
Tasa de entrada en A	0,171	0,0006	unidades de tiempo por petición
Tamaño máximo de la cola en A	363,2	18,8865	peticiones
L de A	117,9008	1,9254	peticiones

Tabla 1.38 Resultados de la prueba de picos de trabajo

Nombre del parámetro	Valor medio obtenido	Varianza obtenida	Unidades
L total	1438,0904	172,0245	peticiones
Tasa de entrada de IN	0,2997	0,0012	unidades de tiempo por petición
Tasa de entrada de IN PICO	0,0998	0,0004	unidades de tiempo por petición
Tasa de llegada a BAL	0,0948	0,0003	unidades de tiempo por petición
Tasa de salida en BAL	0,0499	0,0001	unidades de tiempo por petición
Lambda de BAL	10,5504	0,0323	peticiones por unidad de tiempo
Mu de BAL	20,0456	0,0550	peticiones por unidad de tiempo
Rho de BAL	0,5263	0,0029	-
L de BAL	2,2385	0,0926	peticiones
Lq de BAL	1,7122	0,0901	peticiones
W de BAL	0,2122	0,0082	unidades de tiempo
Wq de BAL	0,1623	0,0081	unidades de tiempo
Tasa de llegada a BAL S	0,0948	0,0003	unidades de tiempo por petición
Tasa de llegada a A	0,1899	0,0009	unidades de tiempo por petición
Tasa de salida en A	0,2002	0,0006	unidades de tiempo por petición
Lambda de A	5,2661	0,0251	peticiones por unidad de tiempo
Mu de A	4,9959	0,0154	peticiones por unidad de tiempo
Rho de A	1,0000	0,0000	-
L de A	1378,8398	170,6100	peticiones
Lq de A	1377,8398	170,6100	peticiones
W de A	262,0840	31,3118	unidades de tiempo
Wq de A	248,3249	28,1456	unidades de tiempo
Tasa de llegada a B	0,1892	0,0003	unidades de tiempo por petición
Tasa de salida en B	0,1996	0,0017	unidades de tiempo por petición
Lambda de B	5,2845	0,0092	peticiones por unidad de tiempo
Mu de B	5,0111	0,0416	peticiones por unidad de tiempo
Rho de B	0,7893	0,0041	-
L de B	2,7775	0,0195	peticiones
Lq de B	1,9882	0,0155	peticiones
W de B	0,7023	0,0071	unidades de tiempo
Wq de B	0,3763	0,0033	unidades de tiempo
Tasa de llegada a C	0,5636	0,0062	unidades de tiempo por petición
Tasa de salida en C	0,1007	0,0009	unidades de tiempo por petición
Lambda de C	1,7746	0,0194	peticiones por unidad de tiempo
Mu de C	9,9319	0,0835	peticiones por unidad de tiempo
Rho de C	0,1787	0,0030	-
L de C	0,5444	0,0127	peticiones
Lq de C	0,3662	0,0122	peticiones
W de C	0,3075	0,0082	unidades de tiempo
Wq de C	0,2068	0,0079	unidades de tiempo
Tasa de llegada a Split P	0,0973	0,0001	unidades de tiempo por petición
Tasa de llegada a Split C	0,5636	0,0062	unidades de tiempo por petición
Tasa de llegada a LOG	0,1299	0,0004	unidades de tiempo por petición
Tasa de salida en LOG	0,1250	0,0000	unidades de tiempo por petición
Lambda de LOG	7,7012	0,0240	peticiones por unidad de tiempo
Mu de LOG	8,0000	0,0000	peticiones por unidad de tiempo
Rho de LOG	0,9621	0,0031	-
L de LOG	53,6901	3,0695	peticiones
Lq de LOG	52,7280	3,0665	peticiones
W de LOG	6,9725	0,3777	unidades de tiempo
Wq de LOG	6,8439	0,3781	unidades de tiempo
Tasa de llegada a LOG S	0,1299	0,0004	unidades de tiempo por petición
W medio total de OUT	206,9338	23,3731	unidades de tiempo

Tabla 1.39 Resultados de la prueba del sistema complejo no analizable por modelos matemáticos simples


UAM
UNIVERSIDAD AUTONOMA
DE MADRID